



IBM Systems - iSeries

ネットワーキング
暗号化ハードウェア

バージョン 5 リリース 4





IBM Systems - iSeries

**ネットワーキング
暗号化ハードウェア**

バージョン 5 リリース 4

ご注意

本書および本書で紹介する製品をご使用になる前に、293 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、IBM i5/OS (プロダクト番号 5722-SS1) バージョン 5、リリース 4、モディフィケーション 0 に適用されます。また、改訂版で断りがない限り、それ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。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

<http://www.ibm.com/jp/manuals/main/mail.html>

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

<http://www.ibm.com/jp/manuals/> の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

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： IBM Systems - iSeries
Networking Cryptographic hardware
Version 5 Release 4

発 行： 日本アイ・ピー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 2006.2

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財)日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

© Copyright International Business Machines Corporation 1998, 2006. All rights reserved.

© Copyright IBM Japan 2006

目次

暗号化	1	2058 暗号化アクセラレーター	287
V5R4 の新機能	1	フィーチャー	288
トピックの印刷	2	シナリオ: システム SSL パフォーマンスの向上	288
暗号化の概念	2	状況:	288
4764 および 4758 暗号化コプロセッサ	8	詳細:	289
暗号化ハードウェアの概念	9	前提条件および前提事項:	289
フィーチャー	11	構成ステップ:	289
暗号化コプロセッサのシナリオ	13	2058 暗号化アクセラレーターの計画	290
暗号化コプロセッサの計画	16	2058 暗号化アクセラレーターの構成	290
暗号化コプロセッサの構成	21	暗号化ハードウェアの関連情報	291
暗号化コプロセッサへの移行	99		
暗号化コプロセッサの管理	126	付録. 特記事項	293
暗号化コプロセッサのトラブルシューティング	273	商標	294
暗号化コプロセッサの再初期化	275	資料に関するご使用条件	295
ハードウェア・サービス管理プログラムの使用	282		

暗号化

- | IBM® は高度なセキュリティーを必要とするお客様向けの暗号化ソリューションを提供します。
- | 総合的な暗号化ソリューションは、セキュリティー戦略の成功に重要な役割を果たします。 IBM はデータ保護やトランザクション処理保護のための一連の暗号化ハードウェア・オプション、およびプログラムをカスタマイズするためにお客様が使用できる一連の暗号化サービス API を提供します。
- | システムに暗号化ハードウェアをインストールすることにより、暗号をセキュリティー計画に不可欠のものとすることができます。暗号化ハードウェアの仕組みおよびシステムへのインプリメント方法を確実に理解するために、以下のトピックをお読みください。
- | 注: この情報には、プログラミング例が含まれています。法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。
- | • 『V5R4 の新機能』。このリリースのシステムで使用可能な新機能について説明します。
- | • 『トピックの印刷』。この情報の PDF バージョンへのリンク、およびユーザーが使用可能な印刷オプションの説明を示します。
- | • 『暗号化の概念』。暗号化機能の基本的な知識、およびシステムの暗号化サービスの概要を示します。
- | • 暗号化ハードウェア:
 - | – 『暗号化ハードウェアの概念』では、システムで使用可能な暗号化ハードウェアに関する基本概念をいくつか説明し、サーバーで暗号化を最大限活用するための方法や、暗号化ハードウェアのオプションをよりよく理解できるようにします。
- | • 使用法のシナリオ。暗号化ハードウェアの構成または使用の一例を示しています。
 - | – 暗号化ハードウェアのシナリオ: 暗号化ハードウェアを使用した秘密鍵の保護
 - | – 暗号化ハードウェアのシナリオ: IBM 暗号化コプロセッサーを使用する i5/OS™ アプリケーションの作成
- | • ご使用のサーバーに最適な暗号化ハードウェアを選択するには、以下のページを参照してください。
 - | – 『暗号化コプロセッサー』。IBM 4764 および 4758 暗号化コプロセッサーの計画および構成情報が示されています。
 - | – 『2058 暗号化アクセラレーター』。2058 暗号化アクセラレーターの計画および構成情報が示されています。
- | • 『関連情報』。暗号に関するその他の情報源、および関連の製品情報のサイトを示しています。

V5R4 の新機能

- | 新しい暗号化ソフトウェアおよびハードウェアに関する最新情報、および既存の暗号化ハードウェア・オプションに追加されたフィーチャーに関する最新情報については、このトピックを参照してください。

追加された IBM 4764 暗号化コプロセッサー

- | IBM 4764 暗号化コプロセッサーには IBM の一連の暗号化ハードウェア・オプションが組み込まれています。4764 暗号化コプロセッサーを注文するには、ハードウェア・フィーチャー・コード 4806 を指定します。前提条件、およびこのコプロセッサーを使用できるシステムのリストについては、16 ページの『要件』を参照してください。

5722-AC3 Cryptographic Access Provider の撤廃



i5/OS V5R4M0 オペレーティング・システムが稼働しているシステムでは、5722-AC3 Cryptographic Access Provider Product を使用しなくても、データ暗号化を使用可能にすることができます。5722-AC3 製品は市販が終了し、その機能は i5/OS オペレーティング・システムに統合されています。つまり、この製品は、この情報に記載されたタスクの一部を実行するための前提条件ではありません。

暗号化サービス API の追加機能

i5/OS V5R4M0 暗号化サービス API には鍵管理機能が追加されていて、暗号鍵を安全に保管および処理することができます。詳しくは、「API ドキュメンテーション (API documentation)」を参照してください。

新機能または変更箇所の見分け方

技術上の変更が行なわれた箇所を見分ける上で役立つように、この情報では以下の記号を使用しています。

-  イメージは、新規情報または変更情報の開始位置を示します。
-  イメージは、新規情報または変更情報の終了位置を示します。

このリリースの新機能または変更機能に関するその他の情報については、「iSeries プログラム資料説明書」を参照してください。

トピックの印刷

この情報の PDF を表示および印刷する場合に使用します。


この情報の PDF 版を参照用または印刷用にダウンロードし、表示することができます。「暗号化ハードウェア」PDF (304 ページ) には、V5R4 のシステムでサポートされる IBM 暗号化ハードウェアに関する情報がすべて記載されています。

PDF ファイルの保管

表示用または印刷用の PDF ファイルをワークステーションに保存するには、次のようにします。

1. ブラウザーで PDF を開く (上記のリンクをクリックする)。
2. PDF をローカルに保管するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「保存」をクリックする。

Adobe Reader のダウンロード

PDF ファイルを表示したり印刷したりするには、Adobe Reader が必要です。これは、Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html)  からダウンロードできます。

暗号化の概念

ここでは、暗号化機能の基本的な知識およびサーバーの暗号化サービスの概要を示します。

暗号化

暗号サービスは、データのプライバシーの保護、データ保全性の維持、通信相手の認証、および否認防止 (相手がメッセージを送信したことを否定する場合) などに役立ちます。

基本的な暗号化を使用すると、保管されたデータまたは通信を、関係のない第三者が理解できないようにしながら、情報を保管したり他の人と通信することができます。暗号化は、理解可能なテキスト（非暗号化テキスト）を理解不能なデータ断片（暗号文）に変換します。暗号化解除は、理解不能なデータから理解可能なテキストを復元します。どちらの機能も、数式（アルゴリズム）と秘密のデータ（鍵）を使用します。

暗号アルゴリズム

暗号アルゴリズムには、以下の 2 つのタイプがあります。

1. 秘密鍵アルゴリズム、つまり対称鍵アルゴリズムでは、鍵が、通信する 2 つの当事者間で共有秘密鍵になります。暗号化および暗号化解除の両方に、同じ鍵を使用します。データ暗号化規格 (DES) および Advanced Encryption Standard (AES) は、対称鍵アルゴリズムの例です。

対称鍵アルゴリズムには、以下の 2 つのタイプがあります。

ブロック暗号

ブロック暗号では、実際の暗号化コードが固定サイズのデータ・ブロックに対して作用します。通常、暗号化/暗号化解除操作に対するユーザーのインターフェースでは、低レベル暗号化機能を繰り返し呼び出すことによって、ブロック・サイズよりも長いデータを取り扱います。データの長さがブロック・サイズ境界に満たない場合は、埋め込みを行わなければなりません。

ストリーム暗号

ストリーム暗号はブロック単位で作用するのではなく、一度に 1 ビット（または 1 バイト）のデータを変換します。

2. 公開鍵アルゴリズム (PKA) つまり非対称鍵アルゴリズムでは、鍵のペアが使用されます。一方の鍵である秘密鍵は秘密にされ、誰とも共有されません。もう一方の鍵である公開鍵は秘密ではなく、他人と共用できます。これらの鍵のどちらか一方でデータが暗号化されると、そのデータはもう一方の鍵でしか暗号を解除して復元することができません。この 2 つの鍵は数学的に関連していますが、秘密鍵を公開鍵から派生させることは事実上不可能です。公開鍵アルゴリズムの例としては、RSA アルゴリズムがあります。

公開鍵アルゴリズムは、対称鍵アルゴリズムよりも処理に時間がかかります。通常、アプリケーションでは、公開鍵アルゴリズムを使用して対称鍵の（鍵の配布のための）暗号化、および（デジタル・シグニチャー生成における）ハッシュの暗号化を行います。

鍵と暗号アルゴリズムが一緒になってデータを変換します。サポートされるすべてのアルゴリズムがパブリック・ドメインにあります。したがって、鍵はデータへのアクセスを制御します。データを保護するには、鍵を保護する必要があります。

暗号操作

さまざまな暗号操作で 1 つ以上のアルゴリズムが使用されます。目的に応じて、暗号操作とアルゴリズム（複数可）を選択してください。例えば、データ保全性を維持するためには、MAC（メッセージ確認コード）操作と AES アルゴリズムを使用します。

システムには、暗号化操作をサポートする API セットがいくつか用意されています。詳しくは、このトピックの末尾にある『システム暗号化の概要』を参照してください。

データのプライバシー

データのプライバシー（機密性）保護を目的とした暗号操作により、無許可ユーザーはメッセージを読み取ることができなくなります。データのプライバシー保護には以下の操作が含まれます。

暗号化および暗号化解除

暗号化操作は、暗号アルゴリズムと鍵を使用して、プレーン・テキスト・データを暗号文に変換します。プレーン・テキスト・データを復元するには、暗号化解除操作で同じアルゴリズムと鍵を使用しなければなりません。

暗号化および暗号化解除は、オペレーティング・システムの任意のレベルで実行できます。次の 3 つのレベルがあります。

フィールド・レベルの暗号化

ユーザー・アプリケーションは、フィールド・レベルの暗号化を使用して、明示的に暗号サービスを要求します。鍵の生成、選択、配布、および暗号化するデータは、ユーザー・アプリケーションによって完全に制御されます。

セッション・レベルの暗号化

システムは、セッション層での暗号化を使用して、アプリケーションに代わって暗号サービスを要求します。アプリケーションが、暗号化が行われていることを認識している場合も、していない場合もあります。

リンク・レベルの暗号化

リンク・レベルの暗号化は、プロトコル・スタックの最下層で実行されますが、通常、リンク・レベルの暗号化のための専用のハードウェアで実行されます。

暗号化コプロセッサおよび 2058 暗号化アクセラレーターは、フィールド・レベルの暗号化および Secure Sockets Layer (SSL) セッション確立の暗号化の両方で使用することができます。i5/OS で VPN がサポートされている場合、コプロセッサやアクセラレーターは使用されません。さらに、SNA セッション・レベルの暗号化もサポートされません。

変換 変換操作は、ある鍵を使用して暗号化されたデータを暗号化解除し、それを別の鍵を使用して暗号化します。これは、アプリケーション・プログラム内のプレーン・テキスト・データの露出を避けるために、1 ステップで実行されます。

データの保全性、認証性、否認防止

データを暗号化するということは、データを操作 (例えば、反復、削除、さらには変更など) できないということではありません。データを信頼するためには、データが許可された送信元から送られていて、かつ変更されていないことを知る必要があります。こうした目的のために、別の暗号操作が必要になります。

ハッシュ (メッセージ要約)

暗号ハッシュ操作は、可変長の入力ストリングから固定長出力ストリング (ダイジェストという) を生成する操作です。実際に、ハッシュ関数には以下のような優れた点があります。

- 衝突防止: データの一部が変更されると、別のハッシュが生成されます。
- 一方方向: この関数は不可逆なものです。つまり、あるダイジェストを生成した元のデータを明らかにすることはできません。

これらの特性によって、ハッシュ操作は認証に有用なものとなっています。例えば、あるダイジェストのコピーを保持しておき、後日新しく生成されたダイジェストと比較することができます。それらのダイジェストが同一のものであれば、データは変更されていません。

MAC (メッセージ確認コード)

MAC 操作は、秘密鍵と暗号アルゴリズムを使用して値 (MAC) を生成する操作です。後でこの値を使用することによって、データが変更されていないことを確認することができます。通常、MAC は送信メッセージの最後に追加されます。メッセージの受信側は、送信側と同じ MAC 鍵、およびアルゴリズムを使用して MAC を複製します。受信側の MAC がメッセージと共に送信された MAC と一致すれば、そのデータは変更されていません。

MAC 操作はメッセージの認証には役立ちますが、送信データがプレーン・テキストのままであるため、無許可の読み取り防止にはなりません。データのプライバシーと安全性の両方を保つには、MAC 操作を実行した後に、メッセージ全体を暗号化する必要があります。

HMAC (Hash MAC)

HMAC 操作は、暗号ハッシュ関数と共有秘密鍵を使用して認証値を生成する操作です。使用方法は MAC と同じです。

署名/検証

署名操作は、デジタル署名と呼ばれる認証値を生成する操作です。署名操作は以下のように実行されます。

1. 署名対象のデータがハッシュされ、ダイジェストが生成される。
2. PKA アルゴリズムと秘密鍵を使用してダイジェストが暗号化され、署名が生成される。

検証操作は以下のように実行されます。

1. 送信側の PKA 公開鍵を使用して署名が暗号化解除され、ダイジェスト 1 が生成される。
2. 署名されたデータがハッシュされ、ダイジェスト 2 が生成される。
3. 2 つのダイジェストが同一であれば、署名の有効性が証明される。

送信側のみが秘密鍵を所有するため、理論的には、この操作で送信側も検証されることとなります。しかし、公開鍵が実際に送信側のものであるということを、どのようにすれば受信側で検証できるのかという問題があります。この問題を解決するために、証明書が使用されます。

鍵および乱数の生成

多くのセキュリティー関連関数は乱数の生成を実行します。例えばパスワード・ソルトの生成や、初期設定ベクトルの生成などです。暗号鍵素材の生成も重要な乱数の使用法の一つです。鍵の生成は、すべてのコンピューター・セキュリティー機能の中でも最も重要なものであるとされています。乱数の暗号強度が低いと、関数がアタックにさらされることとなります。

i5/OS オペレーティング・システムには、疑似乱数発生ルーチン (PRNG) が用意されています。PRNG は多くのシステム機能が使用しており、*Cryptographic Services API* セットを使えばアプリケーションでも利用することができます。

PRNG は、疑似乱数生成とシード管理という 2 つの部分から構成されます。疑似乱数生成は、FIPS 186-1 アルゴリズムを使用して実行されます。暗号強度の高い疑似乱数が得られるかどうかは、シードの質に依存します。FIPS 186-1 の鍵およびシードの値は、システム・シード・ダイジェストから取得されます。システムは、システム情報から収集したデータを使用するか、利用できる場合には、暗号化コプロセッサの乱数発生ルーチンを使用して、自動的にシードを生成します。システムの生成したシードがまったく予測不能なものであるということはありません。暗号化コプロセッサが利用できない場合には、システム・シード・ダイジェストに独自のランダム・シードを追加しなければなりません。この作業は、LIC (Licensed Internal Code) をインストール後、可能な限りすみやかに実行する必要があります。

鍵管理

鍵管理とは、暗号鍵を安全に処理および保管することです。鍵管理には、鍵のストレージと検索、鍵の暗号化と変換、および鍵の配布が含まれます。

鍵ストレージ

システムの鍵ストレージには、以下のものがあります。

- 暗号化サービス鍵ストア

- デジタル証明書マネージャーの証明書ストア
- CCA 鍵ストア (暗号化コプロセッサで使用)
- JCE 鍵ストア

鍵を暗号化コプロセッサに保管することもできます。

鍵の暗号化および変換

保護されたシステム環境の外に鍵を送信したり、保管する場合は、事前に鍵を暗号化する必要があります。さらに、システム内では可能なかぎり暗号化形式で鍵を処理し、機密漏れのリスクを軽減する必要があります。暗号化鍵の管理は、通常、階層型の鍵システムを介して実行されます。

- 最上部にマスター・キーがあります。マスター鍵は唯一のクリアな鍵値であり、セキュアな方法で保管する必要があります。
- その他の鍵を暗号化するには、鍵暗号鍵 (KEK) を使用します。通常、KEK は保管された鍵や、別のシステムに送信される鍵を暗号化する場合に使用します。KEK は通常、マスター鍵を使用して暗号化されます。
- データ鍵は、ユーザー・データに直接使用される鍵です (暗号化や MAC など)。データ鍵は KEK またはマスター鍵を使用して暗号化できます。

鍵のさまざまな用法に合わせて、さまざまな形式の鍵が必要になります。例えば、通常、他の送信元から受信した鍵は、内部形式に変換されます。同様に、システムから送信する鍵は、送信する前に標準外部形式に変換されます。鍵の形式には、ASN.1 BER エンコード形式など標準的なものや、暗号化コプロセッサなど暗号サービス・プロバイダーに固有のものがあります。

鍵配布

通常、データ暗号化は対称鍵アルゴリズムを使用して実行されます。対称鍵は非対称鍵アルゴリズムを使用して配布されます。次の例を検討してください。

- **RSA** - 対称鍵を暗号化してから、配布する場合は、RSA 公開鍵を使用します。対称鍵を暗号化解除するには、対応する秘密鍵を使用します。
- **Diffie-Hellman** - 通信の当事者は D-H パラメーターを生成して、交換し、鍵ペアの生成に使用します。公開鍵が交換されると、各当事者は対称鍵を独自に計算することができます。

システム暗号化の概要

暗号サービス・プロバイダー

暗号サービス・プロバイダー (CSP) は、一連の暗号化操作をインプリメントするソフトウェアまたはハードウェアです。複数の CSP がサポートされています。

- 4758 暗号化コプロセッサ
- 4764 暗号化コプロセッサ
- 2058 暗号化アクセラレーター
- i5/OS LIC
- Java™ Cryptography Extension

暗号化 API セット

ユーザー・アプリケーションは SSL、VPN、IPSec、LDAP などの i5/OS 機能を使用して、間接的に暗号化サービスを利用できます。また、次の API を介して暗号化サービスに直接アクセスすることもできます。

- CCA

Common Cryptographic Architecture (CCA) API セットは、暗号化コプロセッサで暗号操作を実行するために用意されています。

- i5/OS 暗号化サービス

i5/OS 暗号化サービス API セットは、LIC 内で、またはオプションとして 2058 暗号化アクセラレーター上で、暗号操作を実行するために用意されています。

- Java 暗号化

Java Cryptography Extension (JCE) は、Java Software Development Kit の標準拡張機能です。

- ネットワーク認証サービス

GSS (Generic Security Services)、Java GSS、および Kerberos API は、認証およびセキュリティー・サービスを提供するネットワーク認証サービスの一部です。これらのサービスには、セッション・レベルの暗号化機能が含まれています。

- i5/OS SSL および JSSE

i5/OS SSL および JSSE は Secure Sockets Layer プロトコルをサポートします。API はセッション・レベルの暗号化機能を提供します。

- SQL

データベース内の情報にアクセスしたり、変更するには、構造化照会言語を使用します。SQL はデータベース・フィールドの暗号化/暗号化解除をサポートします。

次の表に、各ユーザー・インターフェースで使用される CSP を示します。

表 1. 各ユーザー・インターフェースで使用される CSP

CSP API	i5/OS LIC	JCE	4764 および 4758	2058
CCA			X	
i5/OS 暗号化サービス	X			X
Java 暗号化		X		
ネットワーク認証サービス	X	X		
i5/OS SSL および JSSE	X	X	X	X
SQL	X			

関連概念

143 ページの『鍵ストア・ファイルの初期化』

鍵ストア・ファイルは、操作鍵、つまりマスター鍵で暗号化された鍵を保管する データベース・ファイルです。DES 鍵および PKA 鍵のレコードを保持する場合は、ここに記載された情報を参照してください。

8 ページの『4764 および 4758 暗号化コプロセッサ』

IBM は、さまざまなサーバー・モデルで使用可能な 2 つの暗号化コプロセッサを提供しています。

関連情報

デジタル証明書マネージャー

暗号化サービス API セット (Cryptographic Services API set)

証明書ストア

4764 および 4758 暗号化コプロセッサ

IBM は、さまざまなサーバー・モデルで使用可能な 2 つの暗号化コプロセッサを提供しています。

IBM 4764 暗号化コプロセッサは、ハードウェア・フィーチャー・コード 4806 の eServer™ i5 モデルで使用することができます。以下の表は、それぞれのサーバー・モデルがサポートする暗号化コプロセッサの最大数を示したものです。

表 2. サポートされる 4764 暗号化コプロセッサの数

サーバー・モデル	サーバーごとの最大数	パーティションごとの最大数
eServer i5 モデル 570 8/12/16W、595	32	8
eServer i5 モデル 520、550、570 2/4W	8	8

IBM 4758-023 暗号化コプロセッサは、ハードウェア・フィーチャー・コード 4801 の eServer i5 サーバーで使用することができます。ご使用のサーバーのモデルに応じて、以下の暗号化コプロセッサ数がサポートされています：

表 3. サポートされる 4758 暗号化コプロセッサの数

サーバー・モデル	サーバーごとの最大数	パーティションごとの最大数
eServer モデル 840、870、890 および eServer i5 モデル 570 8/12/16W、595	32	8
eServer モデル 810、820、825、830 および eServer i5 モデル 520、550、570 2/4W	8	8
eServer i5 モデル 800	4	4
eServer i5 モデル 270	3	3

暗号化コプロセッサを使用することで、お客様のサーバーを以下のように増強することができます。

- 暗号化コプロセッサを使用して、さまざまな i5/OS ベースのアプリケーションをインプリメントすることができます。それらのアプリケーションの例としては、会計関連の PIN トランザクション、bank-to-clearing-house トランザクション、IC (チップ) ベースのクレジット・カード用 EMV トランザクション、および基本的な SET™ ブロック処理を実行するアプリケーションなどがあります。これを行うには、ユーザーまたはアプリケーションの提供者が、暗号化コプロセッサのセキュリティー・サービスにアクセスするためのセキュリティー・プログラミング・インターフェース (SAPI) を使用してアプリケーション・プログラムを作成しなければなりません。暗号化コプロセッサの SAPI は、IBM の Common Cryptographic Architecture (CCA) に準拠しています。この SAPI は、i5/OS オプション 35 として納入される CCA 暗号サービス・プロバイダー (CCA CSP) の中に含まれています。

キャパシティーおよび可用性の要件を満たすため、1 つのアプリケーションで最大 8 つのコプロセッサを制御できます。アプリケーションは、Cryptographic_Resource_Allocate (CSUACRA) CCA API および Cryptographic_Resource_Deallocate (CSUACRD) CCA API を使用して、個々のコプロセッサへのアクセスを制御しなければなりません。

- | • 暗号化コプロセッサを DCM と共に使用すると、SSL デジタル証明書に関連付けられた秘密鍵を生成し、それを保管することができます。暗号化コプロセッサでは、SSL セッション確立中に SSL の秘密鍵の処理を操作することにより、パフォーマンス支援の機能拡張が提供されます。
- | • 複数のコプロセッサを使用する場合、DCM 構成では、ハードウェアを使用してデジタル証明書に関連付けられる秘密鍵を生成し、保管するための次のようなオプションが提供されています。
 - | 1. ハードウェアで生成され、ハードウェアで保管された (つまり、保存された) 秘密鍵。このオプションを使用すると、秘密鍵がコプロセッサから出て行くことはないため、その秘密鍵を使用したり、他のコプロセッサと共有することはできません。つまり、システム管理者とアプリケーションは、複数の秘密鍵と証明書を管理しなければなりません。
 - | 2. ハードウェアで生成され、ソフトウェアで保管される (つまり、鍵ストア・ファイルに保管された) 秘密鍵。このオプションを使用すると、1 つの秘密鍵を複数のコプロセッサ間で共有することができます。そのための要件として、各コプロセッサは、同じマスター鍵を共有しなければなりません。「マスター鍵の複製」機能を使用して、同じマスター鍵を使用するコプロセッサをセットアップすることができます。秘密鍵は、コプロセッサの 1 つで生成され、次に鍵ストア・ファイルに保管され、そのコプロセッサのマスター鍵で暗号化されます。同一のマスター鍵を持っていれば、どのコプロセッサでもその秘密鍵を使用することができます。

| 複数の暗号化コプロセッサの管理についての詳細は、『複数の暗号化コプロセッサの管理』を参照してください。

- | • 機能: 暗号化コプロセッサには、i5/OS アプリケーション・プログラムと i5/OS SSL トランザクションが使用する暗号化操作を実行するハードウェア・エンジンが含まれています。各 IBM 暗号化コプロセッサには、マスター鍵を保管するためのセキュア・ストレージを提供する、改ざんされにくいハードウェア・セキュリティー・モジュール (HSM) が含まれています。HSM は、FIPS 140 セキュリティー要件を満たすように設計されています。キャパシティーおよび高可用性に関する要件を満たすため、複数の暗号化コプロセッサがサポートされています。フィーチャー情報は、暗号化コプロセッサと CCA CSP で利用できるものについて詳細に説明しています。
- | • 要件: 暗号化コプロセッサを導入して使用する前に、サーバーはいくつかの要件を満たす必要があります。要件を記載したページを使用して、暗号化コプロセッサをサーバーにインストールして使用できるかどうかを判断してください。
- | • 暗号化の概念: 暗号化についてどの程度知っているかによって、用語または概念についての情報が必要になる場合もあります。このページでは、いくつかの基本的な暗号化概念について紹介します。
- | • 関連情報: IBM が推奨する暗号化についての追加の情報源に関しては、『関連情報』を参照してください。

| 関連概念

| 2 ページの『暗号化の概念』

| ここでは、暗号化機能の基本的な知識およびサーバーの暗号化サービスの概要を示します。

| 暗号化ハードウェアの概念

| システムで暗号を最大限に活用する方法および暗号化ハードウェア・オプションについて理解するには、暗号化ハードウェアに関する次の基本概念を参照してください。

| 注: これらの概念は、IBM 2058 暗号化アクセラレーター・ハードウェアには関連しません。

| 暗号化コプロセッサに関連する鍵のタイプ

| コプロセッサはさまざまな鍵のタイプを使用します。すべての対称鍵操作に対して、すべての DES 鍵あるいは Triple-DES 鍵を使用できるわけではありません。同様に、すべての非対称鍵操作

に対して、すべての公開鍵アルゴリズム (PKA) 鍵を使用できるわけではありません。コプロセッサが使用する各種鍵タイプのリストを以下に示します。

マスター鍵

この鍵はクリア鍵で、他の鍵によって暗号化されていないことを意味します。コプロセッサは、マスター鍵を使用してすべての操作鍵を暗号化します。コプロセッサは、マスター鍵を改ざんされにくいモジュールに保管します。このマスター鍵は、コプロセッサからは取り出すことはできません。コプロセッサは、改ざんに対しては、マスター鍵とその工場認証を破棄することにより対応します。コプロセッサは、DES 鍵の暗号化用と PKA 鍵の暗号化用にそれぞれ 1 つずつの、2 つのマスター鍵を持っています。

倍長鍵暗号化鍵

コプロセッサは、このタイプの Triple-DES 鍵を使用して他の DES 鍵または Triple-DES 鍵を暗号化、または暗号解除します。鍵暗号化鍵は、通常、システム間で鍵を転送するために使用されます。ただし、バックアップ用に、オフラインで鍵を保管するために使用することもできます。鍵暗号化鍵を鍵の転送に使用する場合は、鍵暗号化鍵自体のクリア値を 2 つのシステム間で共有しなければなりません。エクスポートの鍵暗号化鍵は、エクスポート操作に使用されます。エクスポート操作では、マスター鍵で暗号化された鍵は、暗号解除され、次に鍵暗号化鍵で暗号化されます。インポートの鍵暗号化鍵は、インポート操作に使用されます。インポート操作では、鍵暗号化鍵で暗号化された鍵は、暗号解除され、次にマスター鍵で暗号化されます。

倍長 PIN 鍵

コプロセッサはこのタイプの鍵を使用して、金融操作で使用される PIN を生成、検証、暗号化、および暗号解除を行います。これらの鍵は Triple-DES 鍵です。

MAC 鍵

コプロセッサは、このタイプの鍵を使用して、メッセージ確認コード (MAC) を生成します。これらの鍵は、DES 鍵または Triple-DES 鍵のいずれかです。

暗号鍵 コプロセッサは、このタイプの鍵を使用して、データの暗号化または暗号解除を行います。これらの鍵は、DES 鍵または Triple-DES 鍵のいずれかです。

単一長互換性鍵

コプロセッサは、このタイプの鍵を使用してデータの暗号化または暗号解除を行い、さらに MAC を生成します。これらの鍵は DES 鍵であり、Common Cryptographic Architecture をインプリメントしていないシステム間で、暗号化されたデータまたは MAC を交換する場合によく使用されます。

秘密鍵 コプロセッサは、デジタル証明書の生成、および公開鍵で暗号化された DES 鍵または Triple-DES 鍵の暗号解除に秘密鍵を使用します。

公開鍵 コプロセッサは、デジタル署名の妥当性検査、DES 鍵または Triple-DES 鍵の暗号化、および秘密鍵で暗号化されたデータの暗号解除に公開鍵を使用します。

鍵形式 コプロセッサは、4 つの異なる形式のうちいずれか 1 つで、鍵を操作します。鍵形式は、鍵タイプと一緒にあって、暗号プロセスがその鍵を使用する方法を決定します。以下に 4 つの形式を示します。

クリア形式

鍵のクリア値は、どの暗号手段によっても保護されません。クリア鍵は、コプロセッサでは使用できません。クリア鍵は、最初にセキュア・モジュールにインポートして、マスター鍵で暗号化し、次にセキュア・モジュールの外部に保管しなければなりません。

操作可能形式

マスター鍵で暗号化された鍵の形式は、操作可能形式です。これらの鍵は、コプロセッサでは暗号操作に直接使用できます。操作鍵は、内部鍵とも呼ばれます。サーバーの鍵ストア・ファイルに保管されている鍵はすべて操作鍵です。しかし、すべての操作鍵を鍵ストア・ファイルに保管する必要はありません。

エクスポート形式

エクスポート操作の結果、エクスポーターの鍵暗号化鍵で暗号化された鍵の形式は、エクスポート形式です。これらの鍵は外部鍵とも呼ばれます。エクスポート形式の鍵は、エクスポーターの鍵暗号化鍵と同じクリア鍵値を持つインポーターの鍵暗号化鍵が存在している場合は、インポート形式であると記述することもできます。鍵は、任意の方法で、エクスポート形式で保管することができますが、鍵ストア・ファイルに保管することはできません。

インポート形式

インポーターの鍵暗号化鍵で暗号化された鍵の形式は、インポート形式です。インポート操作にソースとして使用できるのは、インポート形式の鍵のみです。これらの鍵は外部鍵とも呼ばれます。インポート形式の鍵は、インポーターの鍵暗号化鍵と同じクリア鍵値を持つエクスポーターの鍵暗号化鍵が存在している場合は、エクスポート形式であると記述することもできます。鍵は、任意の方法で、インポート形式で保管することができますが、鍵ストア・ファイルに保管することはできません。

機能制御ベクトル

IBM は、機能制御ベクトルとして知られる、デジタル署名された値を提供しています。この値を使用すると、コプロセッサ内の暗号アプリケーションは、適用可能なインポート制限とエクスポート制限に一致するレベルの暗号サービスを提供することができます。機能制御ベクトルを使用すると、コプロセッサに鍵の作成に必要な鍵長の情報を得ることができます。

制御ベクトル

制御ベクトルは、機能制御ベクトルとは異なり、以下を制御する鍵に関連付けられた既知の値です。

- 鍵の型
- この鍵が暗号化できる他の鍵の種類
- ユーザーのコプロセッサがこの鍵をエクスポートできるかどうか
- この鍵に対して許可された他の使用

制御ベクトルは暗号を介して鍵にリンクしているため、制御ベクトルを変更する場合は、同時に鍵の値も変更しなければなりません。

鍵ストア・ファイル

コプロセッサのマスター鍵で暗号化された鍵を保管するために使用される、i5/OS データベース・ファイルです。

鍵トークン

暗号鍵、制御ベクトルおよび鍵に関連するその他の情報を持つことができるデータ構造。鍵トークンは、鍵に作用する、あるいは鍵を使用するほとんどの CCA API のパラメーターとして使用されます。

フィーチャー

暗号化コプロセッサには、暗号処理機能および暗号鍵を安全に保管するための手段が備わっています。サポートされる暗号機能には、データの機密を保持するための暗号化、データが変更されていないことを確認

1 するためのメッセージ要約やメッセージ確認コード、そしてデジタル・シグニチャーの生成と検査などが
1 あります。また、コプロセッサには、会計関連の PIN、EMV、および SET アプリケーションに対する
1 充実した基本サービスのセットが用意されています。

1 IBM 4758 および 4764 暗号化コプロセッサ

1 IBM 暗号化コプロセッサの主な利点は、暗号機能を実行し、暗号鍵を管理するためのセキュアな環境が
1 得られることです。マスター鍵は、バッテリー・バックアップ式の改ざんされにくいハードウェア・セキュ
1 リティー・モジュール (HSM) に保管されます。この HSM は、連邦情報処理標準 (FIPS) PUB 140 のセ
1 キュリティー要件を満たすように設計されています。

1 コプロセッサは、ユーザーまたはアプリケーションの提供者によって作成された i5/OS SSL アプリケー
1 ション・プログラムまたは i5/OS アプリケーション・プログラムで使用することができます。4764 暗号
1 化コプロセッサは、4758 暗号化コプロセッサを上回るパフォーマンスを実現します。

1 SSL アプリケーションのフィーチャー

1 Secure Sockets Layer (SSL) またはトランスポート層セキュリティ (TLS) セッションを確立するには、複
1 雑な計算を要する暗号処理が必要になります。i5/OS SSL で 4764 暗号化コプロセッサを使用すると、
1 この複雑な暗号処理の負荷を軽減し、サーバーの CPU をアプリケーション処理に対して解放することがで
1 きます。また、サーバーの SSL デジタル証明書に関連した秘密鍵を、ハードウェア・ベースで保護する
1 こともできます。

1 SSL で構成した場合には、暗号化コプロセッサを使用して、FIPS 140 で認証された HSM に秘密鍵を作
1 成し、保管することができます。また、暗号化コプロセッサを使用すれば、秘密鍵を作成して、マスター
1 鍵を使ってそれを暗号化し (すべて HSM 内で実行)、その暗号化された秘密鍵をシステム・ソフトウェア
1 を使用して鍵ストア・ファイルに保管することができます。これにより、特定の秘密鍵を複数の暗号化コ
1 プロセッサ・カードで使用できるようになります。マスター鍵は常に、FIPS 140 で認証済みのハードウェ
1 ア・モジュールに保管されます。

1 i5/OS CCA アプリケーションのフィーチャー

1 暗号化コプロセッサを使用することで、アプリケーションに高水準の暗号セキュリティを提供すること
1 ができます。暗号化コプロセッサの機能を使用して i5/OS アプリケーションをインプリメントするに
1 は、ユーザーまたはアプリケーション提供者が、暗号化コプロセッサのセキュリティ・サービスにアク
1 セスするためのセキュリティ・アプリケーション・プログラミング・インターフェース (SAPI) を使用し
1 てアプリケーション・プログラムを作成しなければなりません。暗号化コプロセッサの SAPI は、i5/OS
1 Option 35 CCA 暗号サービス・プロバイダー (CCA CSP) が提供する IBM の Common Cryptographic
1 Architecture (CCA) に準拠しています。

1 i5/OS の場合、暗号化コプロセッサの SAPI は、ILE C、RPG、および COBOL で作成されたアプリケ
1 ション・ソフトウェアをサポートします。SAPI を使用したアプリケーション・ソフトウェアは、CCA
1 サービスを呼び出して、さまざまな暗号機能 (Triple-Data Encryption Standard (T-DES)、RSA、MD5、
1 SHA-1、および RIPEMD-160 の各アルゴリズムなど) を実行することができます。会計関連の PIN、
1 EMV2000 (Europay、MasterCard、VISA) 規格、および SET (Secure Electronic Transaction) ブロック処理
1 をサポートする基本サービスも使用できます。セキュリティのオプション層のサポートにおいて、暗号
1 化コプロセッサは役割ベースのアクセス制御機能を備えています。これにより、ユーザーはコプロセッサ
1 がサポートする個々の暗号操作に対するアクセスの実行と制御ができるようになります。役割ベースのア
1 クセス制御はユーザーに与えるアクセス水準を定義します。

1 SAPI は、コプロセッサの鍵管理機能へのアクセスにも使用されます。鍵暗号鍵およびデータ暗号化鍵を
1 定義できます。これらの鍵は暗号化コプロセッサで生成された後、マスター鍵を使用して暗号化されるた
1 め、コプロセッサの外部に保管することができます。ユーザーは、これらの鍵を i5/OS データベース・
1 ファイルである鍵ストア・ファイルに保管します。そのほかの鍵管理機能には以下のようなものがあ
1 ります。

- 1 • 暗号によって保護された乱数生成プログラムを使用しての鍵の作成。
- 1 • 暗号化された T-DES 鍵および RSA 鍵の安全なインポートとエクスポート。
- 1 • マスター鍵の複製の安全な作成。

1 パフォーマンス能力および/または高可用性に関する要件を満たすため、複数の暗号化コプロセッサ・カ
1 ードを使用することができます。詳しくは、『複数の暗号化コプロセッサの管理』を参照してください。

1 4758 および 4764 暗号化コプロセッサのセキュリティー API に関する説明は、「IBM PCI 暗号化コプ
1 ロセッサ CCA 基本サービスのリファレンスおよびガイド リリース 3.23 (IBM PCI Cryptographic
1 Coprocessor CCA Basic Services Reference and Guide, Release 3.23)」に記載されています。これらの資料
1 およびその他の資料は、「IBM PCI Cryptographic Coprocessor documentation library」内で検索できます。

1 暗号化コプロセッサのシナリオ

1 ご使用のシステムでこの暗号化ハードウェアを使用する方法について、いくつかの例を挙げて説明します。

1 シナリオ: 暗号化ハードウェアを使用した秘密鍵の保護

1 このシナリオは、SSL セキュア・ビジネス・トランザクションに関連付けられている、システム・ディジ
1 タル証明書の秘密鍵のセキュリティーを強化する必要がある会社にとって役立ちます。

1 状況

1 ある会社が、企業間取引 (B2B) トランザクションを処理するための専用のシステムを使用しています。こ
1 の会社のシステムのスペシャリストである Sam は、上司から B2B カスタマーのセキュリティー要件を知
1 らされています。その要件とは、Sam の会社が実行するシステムの SSL セキュア・ビジネス・トランザク
1 ションに関連付けられている、デジタル証明書の秘密鍵のセキュリティーを強化することです。Sam
1 は、暗号化コプロセッサ・カードという、システムで使用可能な暗号化ハードウェア・オプションがあ
1 り、その改ざんされにくいハードウェアで SSL トランザクションに関連付けられている秘密鍵を暗号化
1 および保管できることを知ります。

1 Sam は暗号化コプロセッサについて調べ、このコプロセッサを i5/OS デジタル証明書マネージャー
1 (DCM) と一緒に使用するとセキュアな SSL 秘密鍵の保管場所を提供できること、さらに、SSL のセッシ
1 ョン確立時に実行される暗号操作をシステムからオフロードすることでシステムのパフォーマンスが向上
1 することを知ります。

1 注: ロード・バランシングとパフォーマンス・スケーリングをサポートするために、Sam は、システムの
1 SSL で複数の 4758 暗号化コプロセッサを使用することができます。

1 Sam は、自社のシステムのセキュリティーを強化するという要件を、暗号化コプロセッサが満たすと判
1 断します。

1 関連概念

1 184 ページの『複数の暗号化コプロセッサの管理』
1 区画ごとに最大 8 つの暗号化コプロセッサを持つことができます。サーバーごとにサポートされる
1 暗号化コプロセッサの最大数は、システム・モードによって異なります。複数のコプロセッサを
1 SSL で使用する場合は、このトピックを参照してください。

詳細

1. この会社のシステムに暗号化コプロセッサがインストールされ、秘密鍵を保管して保護するように構成されています。
2. 暗号化コプロセッサによって、秘密鍵が生成されます。
3. その後、秘密鍵が暗号化コプロセッサに保管されます。
4. 暗号化コプロセッサが、物理的および電子的な両方のハッキングからサーバーを保護します。

前提条件および前提事項

1. システムに暗号化コプロセッサがインストールされ、適切に構成されています。暗号化コプロセッサの計画には、システムで SSL を実行することも含まれます。

注: アプリケーションの SSL ハンドシェイク処理および秘密鍵の保護のために複数の暗号化コプロセッサ・カードを使用するには、Sam は、アプリケーションが複数の秘密鍵および証明書を管理できることを確認する必要があります。

2. Sam の会社ではデジタル証明書マネージャー (DCM) がインストールされ、構成されており、これを使用して、SSL 通信セッションのための公衆インターネット証明書の管理を行なっています。
3. Sam の会社は、公衆認証局 (CA) から証明書を取得しています。
4. 暗号化コプロセッサは、DCM を使用する前にオンにしておきます。これを行わない場合、DCM は、証明書作成プロセスの一部としてストレージ・オプションを選択するためのページを提供しません。

関連概念

16 ページの『暗号化コプロセッサの計画』

以下の情報は、サーバーへの IBM 暗号化コプロセッサのインストールを計画する担当者を対象としたものです。

21 ページの『暗号化コプロセッサの構成』

暗号化コプロセッサを構成すると、その暗号操作をすべて使用できるようになります。

関連情報

SSL 通信セッションのための公開インターネット証明書の管理

構成ステップ

システムで暗号化ハードウェアを使用して秘密鍵を保護するためには、Sam は以下のステップを実行する必要があります。

1. このシナリオの前提条件および前提事項が満たされていることを確認します。
2. IBM デジタル証明書マネージャー (DCM) を使用して、新しいデジタル証明書を作成するか、あるいは現在のデジタル証明書を更新します。
 - a. 現在の証明書に署名した認証局 (CA) のタイプを選択します。
 - b. 証明書の秘密鍵のストレージ・オプションとして、「ハードウェア」を選択します。
 - c. 証明書の秘密鍵を保管する、暗号化ハードウェア装置を選択します。
 - d. 公衆 CA を使用するよう選択します。

これで、新しいデジタル証明書に関連付けられた秘密鍵は、ステップ 2.c で指定した暗号化コプロセッサに保管されます。Sam はここで、自社の Web サーバーの構成に進むことができ、新しく作成された証明書を使用するよう指定することができます。Web サーバーを再始動した後、Web サーバーは新しい証明書を使用します。

関連概念

99 ページの『i5/OS アプリケーションで使用するための暗号化コプロセッサの構成』
このトピックでは、暗号化コプロセッサを i5/OS アプリケーションで使用できるようにするために必要なステップをリストしています。

シナリオ: 暗号化コプロセッサを使用する i5/OS アプリケーションの作成

このシナリオは、暗号化コプロセッサを呼び出して、ユーザー・データ (自動預金支払機 (ATM) で入力された、金融サービスにおける個人識別番号 (PIN) など) を検証するプログラムを作成するプロセスを通じて、i5/OS プログラマーにとって役立ちます。

関連概念

18 ページの『セキュア・アクセス』
アクセス制御は、システム資源の使用を、資源との対話を許可されているユーザーのみに制限します。サーバーでは、システム資源へのユーザーの権限を管理することができます。

21 ページの『暗号化コプロセッサの構成』
暗号化コプロセッサを構成すると、その暗号操作をすべて使用できるようになります。

状況

あなたが、大規模な金融関連の Credit Union 社のシステム・プログラマーであるとして、Credit Union 社のシステムにインストールされている暗号化コプロセッサ PCI カードを使用して、メンバーの金融関連個人識別番号 (PIN) が自動預金支払機 (ATM) で入力された際にこの PIN を検査するという作業を任せられました。


暗号化コプロセッサの暗号サービスにアクセスしてメンバーの PIN を検査するために、オプション 35 の一部分である CCA CSP (Cryptographic Service Provider) API を使用する、i5/OS アプリケーション・プログラムを作成することに決めました。暗号化コプロセッサ用に作成された i5/OS アプリケーション・プログラムは、コプロセッサを利用して、セキュリティー依存タスクおよび暗号化処理を実行します。

注: CCA CSP を介して複数の暗号化コプロセッサを使用できます。アプリケーションは、Cryptographic_Resource_Allocate (CSUACRA) CCA API および Cryptographic_Resource_Deallocate (CSUACRD) CCA API を使用して、個々のコプロセッサへのアクセスを制御しなければなりません。

詳細

1. Credit Union 社のメンバーが、ATM で自分の PIN を入力します。
2. PIN は ATM で暗号化され、その後、ネットワークを経由して Credit Union 社のシステムに送信されます。
3. システムがそのトランザクション要求を認識し、そのメンバーの PIN を検査するプログラムを呼び出します。
4. プログラムが、暗号化された PIN、メンバーの口座番号、PIN 生成鍵、および PIN 暗号鍵を含む要求を暗号化コプロセッサに送信します。
5. 暗号化コプロセッサが、PIN の妥当性を確認または拒否します。
6. プログラムが、暗号化コプロセッサからの結果を ATM に送信します。
 - a. PIN が確認された場合、メンバーは Credit Union 社との取引を正常に実行できます。
 - b. PIN が拒否された場合、メンバーは Credit Union 社との取引を実行できません。

前提条件および前提事項:

1. あなたの会社にシステムがあり、そこに暗号化コプロセッサが正しくインストールされ、構成されていることが前提です。以下の情報を参照してください。
 - a. 暗号化コプロセッサの計画
 - b. 暗号化コプロセッサの構成
 - c. i5/OS アプリケーションで使用するための暗号化コプロセッサの構成
2. オプション 35 の Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP) を十分に理解していることが必要です。これは i5/OS オプション 35 としてパッケージされており、暗号化コプロセッサの暗号サービスへのアクセスを可能とするアプリケーションを作成することができる、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 (SAPI) を提供します。
3. 「CCA Basic Services Guide」  にアクセス可能である必要があります。ここには、アプリケーションで使用する Financial Services Support の verb が記載されています。

構成ステップ

暗号化コプロセッサを使用して PIN の妥当性を検査するという目標を達成する方法の 1 つとして、サーバーの 2 つの i5/OS アプリケーションを作成する方法があります。

1. PIN 検査鍵と PIN 暗号鍵の両方をロードし、それらを鍵ストア・ファイルに保管するプログラムを作成します。クリア鍵パーツが使用されると想定した場合、以下の API を使用する必要があります。
 - Logon_Control (CSUALCT)
 - Key_Part_Import (CSNBKPI)
 - Key_Token_Build (CSNBKTB)
 - Key_Record_Create (CSNBKRC)
 - Key_Record_Write (CSNBKRW)
 - オプション API: KeyStore_Designate (CSUAKSD)
2. Encrypted_PIN_Verify (CSNBPVR) API を呼び出して、暗号化された PIN を検査し、その後で有効であるか無効であるかという状況を ATM に戻す、2 番目のプログラムを作成します。

暗号化コプロセッサの計画

以下の情報は、サーバーへの IBM 暗号化コプロセッサのインストールを計画する担当者を対象としたものです。

インストール前の準備

暗号化コプロセッサをインストールする前に、ご使用のサーバーが暗号化コプロセッサの要件を満たしているかどうかを確認することが重要です。これらの要件には、ハードウェアおよびソフトウェアの前提条件が含まれます。また、暗号化コプロセッサをインストールする前に、サーバーのリソースに安全にアクセスできるかどうかを確認する必要があります。最後に、セキュリティー API (SAPI) に必要なオブジェクト権限に精通する必要があります。

- 要件
- セキュア・アクセス
- SAPI に必要なオブジェクト権限

要件

コプロセッサのいずれかを導入して使用する前に、システムが以下の要件を満たす必要があります。

4764 暗号化コプロセッサの要件

4764 暗号化コプロセッサ (ハードウェア・フィーチャー・コード 4806 で注文できます) は、以下の System i5 モデルでサポートされています。

- iSeries™ i5 520、550、570、および 595
- I/O 拡張タワー 5074、5088、5095、0595、5294、および 5790

暗号化コプロセッサは PCI カードであり、以下のソフトウェアが必要です。

- i5/OS (5722-SS1): 4764 暗号化コプロセッサには、i5/OS バージョン 5 リリース 3 モディフィケーション 0 以降が必要です。

注: V5R3M0 が稼働しているシステムの場合、ハードウェアで暗号機能を使用可能にするには、Cryptographic Access Provider 128-bit (5722-AC3) ライセンス・プログラム製品もインストールされなければなりません。

- i5/OS Option 35 Common Cryptographic Architecture 暗号サービス・プロバイダー (CCA CSP) は、SAPI を提供します。
- i5/OS 5733-CY1 Cryptographic Device Manager は、4764 暗号化コプロセッサ用の CCA ファームウェアを提供します。
- i5/OS オプション 34 デジタル証明書マネージャー (暗号化コプロセッサ構成のための Web ベースのユーティリティを使用する予定の場合)
- i5/OS 5722-TC1 TCP/IP 接続ユーティリティ (暗号化コプロセッサ構成のための Web ベースのユーティリティを使用する予定の場合)
- i5/OS 5722-DG1 IBM HTTP Server (暗号化コプロセッサ構成のための Web ベースのユーティリティを使用する予定の場合)

4758 暗号化コプロセッサの要件

4758 暗号化コプロセッサ (ハードウェア・フィーチャー・コード 4801 で注文できます) は、以下のサーバー・モデルでサポートされています。

- iSeries i5 520、550、570、および 595
- iSeries 250 および 270 (250 では 7102 拡張装置が必要)、810、820、825、830、840、870、および 890
- 拡張装置 5074、5075、5078、5079、5088、5094、5095、5294 および 5790

暗号化コプロセッサは PCI カードです。eServer PCI アダプター・インストール・マニュアルの説明に従って、このカードをインストールしてください。詳しくは、PCI アダプターの PDF を参照してください。

暗号化コプロセッサには、以下のソフトウェアが必要です。

- i5/OS: 4758 暗号化コプロセッサには、i5/OS バージョン 5 リリース 3 モディフィケーション 0 以降が必要です。

注: V5R3M0 が稼働しているシステムの場合、ハードウェアで暗号機能を使用可能にするには、Cryptographic Access Provider 128-bit (5722-AC3) ライセンス・プログラム製品もインストールされなければなりません。

- i5/OS Option 35 Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP)。
- i5/OS オプション 34 デジタル証明書マネージャー (暗号化コプロセッサ構成のための Web ベースのユーティリティを使用する予定の場合)

l • i5/OS 5722-TC1 TCP/IP 接続ユーティリティー (暗号化コプロセッサ構成のための Web ベースのユーティリティーを使用する予定の場合)

l • i5/OS 5722-DG1 IBM HTTP Server (暗号化コプロセッサ構成のための Web ベースのユーティリティーを使用する予定の場合)

l **ソフトウェアに関する注記:** i5/OS V5R4M0 が稼働している場合、5722-AC3 Cryptographic Access Provider 製品を使用しなくても、データ暗号化を使用可能にすることができます。5722-AC3 製品は市販が終了し、その機能は i5/OS に統合されています。i5/OS V5R4M0 をインストールすると、以前にインストールされていたすべての 5722-ACx 製品が自動的に削除されます。

l **ハードウェアに関する注記:** -15 度 C (5 度 F) 未満に冷却されると、暗号化コプロセッサの工場認証は破棄されます。コプロセッサの工場認証が無効になると、カードは使えなくなり、新規暗号化コプロセッサを発注する場合は、ハードウェア・サービス・プロバイダーに問い合わせなければなりません。

l セキュア・アクセス

l アクセス制御は、システム資源の使用を、資源との対話を許可されているユーザーのみに制限します。サーバーでは、システム資源へのユーザーの権限を管理することができます。

l ユーザーの組織は、組織のセキュリティ階層にある各システム資源を識別する必要があります。階層は、ユーザーが資源に対して所有するアクセス権限のレベルを明確に概説する必要があります。

l i5/OS オプション 35 のサービス・プログラムはすべて、*PUBLIC に対して *EXCLUDE 権限を指定して出荷されています。この場合、サービス・プログラムを使用するための *USE 権限をユーザーに与える必要があります。さらに、ユーザーには、ライブラリー QCCA にある QC6SRV サービス・プログラムへの *USE 権限も与える必要があります。

l 暗号化コプロセッサのセットアップを行うユーザーは、Master_Key_Process (CSNBMKP)、Access_Control_Initialize (CSUAACI)、または Cryptographic_Facility_Control (CSUACFC) のセキュリティ・アプリケーション・プログラミング・インターフェース (SAPI) を使用するための *IOSYSCFG 特殊権限を持っている必要があります。これら 3 つの SAPI は、暗号化コプロセッサのすべての構成ステップを実行するために使用されます。すべての SAPI に対して、ユーザーが追加オブジェクト権限を必要とする場合もあります。

l 非常に保護機能の高い環境の場合は、*ALLOBJ 特殊権限を持っていないユーザーにコプロセッサ管理者の役割を割り当てることを考慮してください。割り当てた場合、*ALLOBJ 特殊権限を持っているユーザーは、コプロセッサ上の管理役割にログオンできないため、コプロセッサの構成を変更することはできません。ただし、SAPI サービス・プログラムへのオブジェクト権限を制御して、管理者による誤用を防ぐことができます。

l 暗号化コプロセッサ構成のための Web ユーティリティーを使用するには、ユーザーは *SECADM 特殊権限をもっていなければなりません。

l 暗号化コプロセッサは、サーバーのアクセス制御に関連しない別個のアクセス制御権を持っています。暗号化コプロセッサ・アクセス制御を使用すると、暗号化コプロセッサ・ハードウェア・コマンドへのアクセスを制御することができます。

l セキュリティーをさらに強化するには、暗号化コプロセッサ内部のデフォルトの役割の機能を制限します。マスター鍵の変更のような、機密性の高い機能を実行するためには、各役割間で 2 人以上を必要とするように、機能を割り当てます。この操作は、役割とプロファイルを操作する際に実行できます。

注: 同様に、サーバーを施錠されたドアの背後に保管するような、標準的な物理的セキュリティ手段を考慮する必要があります。

関連概念

24 ページの『役割およびプロファイルの作成と定義』

暗号化コプロセッサは、役割ベースのアクセス制御を使用します。役割ベースのシステムでは、コプロセッサ・ユーザーのクラスに対応する一連の役割を定義します。使用可能な役割の 1 つにユーザーをマップするために、関連したユーザー・プロファイルを定義して各ユーザーを登録できます。

98 ページの『DCM および SSL で使用するための暗号化コプロセッサの構成』

暗号化コプロセッサを SSL で使用できるようにする場合は、ここに記載された情報を参照してください。

15 ページの『シナリオ: 暗号化コプロセッサを使用する i5/OS アプリケーションの作成』

このシナリオは、暗号化コプロセッサを呼び出して、ユーザー・データ (自動預金支払機 (ATM) で入力された、金融サービスにおける個人識別番号 (PIN) など) を検証するプログラムを作成するプロセスを通じて、i5/OS プログラマーにとって役立ちます。

関連資料

『SAPI に必要なオブジェクト権限』

SAPI に必要なオブジェクト権限に関する情報については、次の表を参照してください。

SAPI に必要なオブジェクト権限:

SAPI に必要なオブジェクト権限に関する情報については、次の表を参照してください。

SAPI	*USE (装置用)	*USE (DES 鍵ストア用)	*CHANGE (DES 鍵ストア用)	*USE (DES 鍵ストア・ライブラリー用)	*USE (PKA 鍵ストア用)	*CHANGE (PKA 鍵ストア用)	*USE (PKA 鍵ストア・ライブラリー用)
CSNBCKI	Y		Y ¹	Y ¹			
CSNBCKM	Y		Y ¹	Y			
CSNBCPA	Y	Y ¹		Y ¹			
CSNBCPE	Y	Y ¹		Y ¹			
CSNBCSG	Y	Y ¹		Y ¹			
CSNBCSV	Y	Y ¹		Y ¹			
CSNBCVE	Y	Y ¹		Y ¹			
CSNBCVG							
CSNBCVT	Y	Y ¹		Y ¹			
CSNBDEC	Y	Y ¹		Y ¹			
CSNBDKG	Y		Y ¹	Y ¹			
CSNBDKM	Y	Y ²	Y ²	Y ¹			
CSNBDKX	Y	Y ¹		Y ¹			
CSNBENC	Y	Y ¹		Y ¹			
CSNBEPG	Y	Y ¹		Y ¹			
CSNBKET	Y	Y ¹		Y ¹			
CSNBKEX	Y	Y ¹		Y ¹			
CSNBKGN	Y	Y ²	Y ²	Y ¹			
CSNBKPI	Y		Y ¹	Y ¹			

SAPI	*USE (装置用)	*USE (DES 鍵ストア用)	*CHANGE (DES 鍵ストア用)	*USE (DES 鍵ストア・ライブラリー用)	*USE (PKA 鍵ストア用)	*CHANGE (PKA 鍵ストア用)	*USE (PKA 鍵ストア・ライブラリー用)
CSNBKRC	Y		Y	Y			
CSNBKRD	Y		Y	Y			
CSNBKRL	Y	Y		Y			
CSNBKRR	Y	Y		Y			
CSNBKRW	Y		Y	Y			
CSNBKSI	Y		Y ³	Y ³		Y ³	Y ³
CSNBKTC	Y		Y ¹	Y ¹			
CSNBKTP							
CSNBKTR	Y	Y ¹		Y ¹			
CSNBKYT	Y	Y ¹		Y ¹			
CSNBKYTX ⁴	Y	Y ¹		Y ¹			
CSNBMDG	Y						
CSNBMGN	Y	Y ¹		Y ¹			
CSNBMKP	Y						
CSNBOWH							
CSNBPCU	Y	Y ¹		Y ¹			
CSNBPEX	Y	Y ¹		Y ¹			
CSNBPEXX ⁴	Y	Y ¹		Y ¹			
CSNBPGN	Y	Y ¹		Y ¹			
CSNBSPN	Y	Y ¹		Y ¹			
CSNBPTR	Y	Y ¹		Y ¹			
CSNBPVR	Y	Y ¹		Y ¹			
CSNBSKY	Y	Y ¹		Y ¹			
CSNBTRV	Y	Y		Y			
CSNDDSG	Y				Y ¹		Y ¹
CSNDDSV	Y				Y ¹		Y ¹
CSNDKRC						Y	Y
CSNDKRD						Y	Y
CSNDKRL					Y		Y
CSNDKRR					Y		Y
CSNDKRW						Y	Y
CSNDKTC	Y					Y ¹	Y ¹
CSNDPKB							
CSNDPKG	Y	Y ¹				Y ¹	Y ¹
CSNDPKH	Y						
CSNDPKI	Y	Y ¹				Y ¹	Y ¹
CSNDPKR	Y						
CSNDPKX	Y				Y ¹		Y ¹

SAPI	*USE (装置用)	*USE (DES 鍵ストア用)	*CHANGE (DES 鍵ストア用)	*USE (DES 鍵ストア・ライブラリー用)	*USE (PKA 鍵ストア用)	*CHANGE (PKA 鍵ストア用)	*USE (PKA 鍵ストア・ライブラリー用)
CSNDRKD	Y						
CSNDRKL	Y						
CSNDSBC	Y				Y ¹		Y ¹
CSNDSBD	Y				Y ¹		Y ¹
CSNDSYG	Y					Y ¹	Y ¹
CSNDSYI	Y		Y ¹	Y ¹	Y ¹		Y ¹
CSNDSYX	Y		Y ¹	Y ¹	Y ¹		Y ¹
CSUAACI	Y						
CSUAACM	Y						
CSUACFC	Y						
CSUACFQ	Y						
CSUACRA	Y						
CSUACRD	Y						
CSUAKSD							
CSUALCT	Y						
CSUAMKD	Y						

¹ この API に対するデータ暗号化規格 (DES) または公開鍵アルゴリズム (PKA) 鍵ストアの使用は、オプションです。

² オプションとして、複数のパラメーターが鍵ストアを使用することができます。権限要求は、これらのパラメーターごとに異なります。

³ Key_Store_Initialize SAPI は、同時に両方のファイルに対する権限を必要とすることはありません。

⁴ これらの SAPI は、4764 コプロセッサにのみ関係します。

関連概念

18 ページの『セキュア・アクセス』

アクセス制御は、システム資源の使用を、資源との対話を許可されているユーザーのみに制限します。サーバーでは、システム資源へのユーザーの権限を管理することができます。

暗号化コプロセッサの構成

暗号化コプロセッサを構成すると、その暗号操作をすべて使用できるようになります。

最も簡単、かつ迅速に暗号化コプロセッサを構成するには、暗号化コプロセッサ構成のための Web ベースのユーティリティーを使用します。このユーティリティーには、`http://server-name:2001` (お客様がすでにポート番号 2001 から別の番号に変更済みの場合は、その番号を使用して下さい) の「System Tasks」ページからアクセスできます。このユーティリティーには、これまでに構成されていないコプロセッサの構成 (および初期化) に使用される、基本構成ウィザードが含まれています。HTTP と SSL がこれまでに構成されていない場合は、構成ウィザードを使用する前に以下の手順を実行する必要があります。

- HTTP 管理サーバーを開始します。

- | • HTTP 管理サーバーが SSL を使用できるように構成します。
- | • DCM を使用して証明書を作成します。このとき、秘密鍵をソフトウェアで構成して保管することを指定します。
- | • DCM を使用して、署名された証明書を受信します。
- | • その証明書を HTTP 管理サーバーのアプリケーション ID に関連付けます。
- | • HTTP 管理サーバーを再始動して、SSL 処理をできるようにします。

| 暗号化コプロセッサがすでに構成済みの場合は、「構成の管理」オプションをクリックして、コプロセッサの特定の部分についての構成を変更します。

| 独自のアプリケーションを作成して、コプロセッサを構成することもできます。これを行うには、Cryptographic_Facility_Control (CSUACFC)、Access_Control_Initialize (CSUAACI)、Master_Key_Process (CSNBKMP)、および Key_Store_Initialize (CSNBKSI) API verb を使用します。このセクションの多くのページには、アプリケーションによってコプロセッサを構成する方法を示すプログラム例が 1 つ以上含まれています。必要に応じてこれらのプログラムを変更してください。

| 暗号化コプロセッサ構成ユーティリティを使用するか、独自にアプリケーションを作成するかに関係なく、暗号化プロセッサを適切に構成するために必要となるステップを以下に示します。

| 関連概念

| 98 ページの『DCM および SSL で使用するための暗号化コプロセッサの構成』

| 暗号化コプロセッサを SSL で使用できるようにする場合は、ここに記載された情報を参照してください。

| 15 ページの『シナリオ: 暗号化コプロセッサを使用する i5/OS アプリケーションの作成』

| このシナリオは、暗号化コプロセッサを呼び出して、ユーザー・データ (自動預金支払機 (ATM) で入力された、金融サービスにおける個人識別番号 (PIN) など) を検証するプログラムを作成するプロセスを通じて、i5/OS プログラマーにとって役立ちます。

| 装置記述の作成

| 装置記述は、デフォルトの鍵の保管場所を指定します。装置記述は、鍵ストア・ファイルの命名あり、または、なしで作成することができます。

| 暗号化コプロセッサの装置記述をサーバーに作成する必要があります。装置記述は、CCA CSP により使用され、コプロセッサに直接暗号要求する際に役に立ちます。さらに装置記述は、コプロセッサに対して鍵ストア・ファイルのデフォルトの保管場所を示します。暗号化コプロセッサ構成ユーティリティー (<http://server-name:2001> の「System Tasks」ページからアクセスできます) の基本構成ウィザードで、装置記述を作成することができます。あるいは、装置記述の作成 (暗号) (Create Device Crypto) CL コマンドを使用して、独自に装置記述を作成することもできます。

| 基本構成ウィザードを使用して装置記述を作成するには、以下のステップを実行します。

- | 1. Web ブラウザーで <http://server-name:2001> の「System Tasks」ページにアクセスします。
- | 2. 「暗号化コプロセッサの構成」をクリックします。
- | 3. 「セキュア・セッションの開始 (Start secure session)」ラベルのついたボタンをクリックします。
- | 4. 「基本構成」ウィザードをクリックします。
- | 5. 「Welcome」ページで「続行」をクリックします。
- | 6. 使用するリソースに対して *CREATE に設定された装置名を持つリスト項目をクリックします。
- | 7. 基本構成ウィザードの指示に従って続行します。

CL を使用した装置記述の作成:

CL コマンドを使用して装置記述を作成するには、以下のステップを実行します。

1. CL コマンド行に CRTDEVCRP と入力します。
2. プロンプトが表示されたときに、装置の名前を指定します。デフォルトの装置をセットアップする場合には、装置に CRP01 という名前を付けます。デフォルトの装置をセットアップしない場合は、装置記述にアクセスするために、Cryptographic Resource Allocate (CSUACRA) API を使用しなければなりません。
3. デフォルトの PKA 鍵ストア・ファイルの名前を指定するか、パラメーターのデフォルトを *NONE にします。
4. デフォルトの DES 鍵ストア・ファイルの名前を指定するか、パラメーターのデフォルトを *NONE にします。
5. **任意:** プロンプトが表示されたら、記述を指定します。
6. 装置記述の作成が完了した後、構成変更 (VRYCFG) または構成状況処理 (WRKCFGSTS) CL コマンドを使用して、装置の構成を変更します。通常は完了するまでに 1 分ですが、10 分かかることもあります。

注: APPTYPE のデフォルト値は *CCA なので、**Create** コマンド上で指定する必要はありません。ただし、他の値に変更していた場合は、装置をオンに変更するためにはその前に APPTYPE を *CCA に戻す必要があります。

以上で、装置記述の作成が完了しました。

鍵ストア・ファイルへの命名

鍵ストア・ファイルまたは鍵ストア・ファイルに保管されている鍵を使用して何らかの操作を実行する前に、鍵ストア・ファイルに名前を付ける必要があります。

2 つのタイプの鍵ストア・ファイルに名前を付けることができます。1 つのタイプは、データ暗号化規格 (DES) 鍵と Triple-DES 鍵を保管します。DES および Triple-DES は対称暗号アルゴリズムであり、暗号化コプロセッサは同じ鍵を使用して、暗号化と暗号化解除を行います。もう一方のタイプは、公開鍵アルゴリズム (PKA) 鍵を保管します。公開鍵アルゴリズムは非対称で、鍵はペアで作成されます。暗号化コプロセッサは、1 つの鍵を使用して暗号化し、別の鍵を使用して暗号化解除します。暗号化コプロセッサは、RSA 公開鍵アルゴリズムをサポートしています。

プログラムを使用して鍵ストア・ファイルに明示的に名前を付けることもできますが、装置記述上で構成して名前を付けることもできます。プログラムから鍵ストア・ファイルに名前を付けるには、Key_Store_Designate (CSUKSD) セキュリティ・アプリケーション・プログラミング・インターフェース (SAPI) を使用します。プログラムを使用する鍵ストア・ファイルに名前を付ける場合、暗号化コプロセッサは、プログラムを実行したジョブの名前のみを使用します。しかし、鍵ストア・ファイルにプログラムで明示的に名前を付けると、別個の鍵ストア・ファイルを他のユーザーから使用することができるようになります。装置記述上にある鍵ストア・ファイルに名前を付ける場合には、プログラムで名前を付ける必要はありません。これは、複数の IBM プラットフォームで同じプログラム・ソースを維持する場合に役立ちます。Common Cryptographic Architecture の別のインプリメンテーションからプログラムを移植する場合にも役に立ちます。

暗号鍵を長期にわたって使用したり、他のユーザーまたはサーバーと交換できるようにするには、暗号鍵を安全な形式で保管する必要があります。暗号鍵は、ユーザー独自の方法を使用して保管することもできます

が、鍵ストア・ファイルに保管することもできます。鍵ストア・ファイルは必要なだけ持つことができるので、各タイプの鍵に対して複数の鍵ストア・ファイルを作成することができます。鍵ストア・ファイルには、暗号鍵を必要なだけ置くことができます。

各鍵ストア・ファイルは別個のサーバー・オブジェクトであるため、各ファイルごとに異なるユーザーを許可することができます。各鍵ストア・ファイルは、異なる時点で保管したり、回復することができます。ファイル・データが変更される頻度またはどのデータが保護されているかによって違いがあります。

役割およびプロファイルの作成と定義

暗号化コプロセッサは、役割ベースのアクセス制御を使用します。役割ベースのシステムでは、コプロセッサ・ユーザーのクラスに対応する一連の役割を定義します。使用可能な役割の 1 つにユーザーをマップするために、関連したユーザー・プロファイルを定義して各ユーザーを登録できます。

役割の機能は、その役割に対して使用可能になっているアクセス制御ポイントあるいは暗号化ハードウェア・コマンドに依存しています。暗号化コプロセッサを使用すると、ユーザーが選択する役割に基づいたプロファイルを作成することができます。

役割ベースのシステムは、権限がユーザーごとに個別に割り当てられるものよりも効率的です。一般に、ユーザーはアクセス権のカテゴリ別に数種類に分類できます。役割を使用すると、役割の形式で、1 回だけこれらの各カテゴリを定義することができます。

役割ベースのアクセス制御システム、および使用可能な許可コマンドのグループは、各種のセキュリティ・ポリシーをサポートするように設計されています。特に、二重制御の分割ポリシーを強制的に実施するように、暗号化コプロセッサをセットアップすることができます。このポリシーの下では、暗号化コプロセッサが完全にアクティブになった後は、いかなる人物も、サービス妨害の攻撃以外の有害な行為を実行することはできません。このポリシーおよびその他多くのアプローチをインプリメントするには、特定のコマンドの使用を制限する必要があります。アプリケーションを設計する時点で、アクセス制御システムで使用可能にするコマンドあるいは制限するコマンドを検討し、セキュリティ・ポリシーの意味についても検討してください。

すべての暗号化コプロセッサは、デフォルトの役割と呼ばれる役割を持っていない限りなりません。暗号化コプロセッサにログオンしていないユーザーは、デフォルトの役割で定義されている機能を実行することになります。デフォルトの役割で定義された機能のみを必要とするユーザーは、プロファイルは必要ありません。ほとんどのアプリケーションでは、ユーザーの大部分はデフォルトの役割で操作し、ユーザー・プロファイルを使用しません。一般的に、プロファイルを必要とするのは、機密保護担当者と特殊なユーザーのみです。

暗号化コプロセッサが初期化されていない状態にある場合は、デフォルトの役割では、以下のアクセス制御ポイントが使用可能になっています。

- PKA96 片方向ハッシュ
- クロックの設定
- 再初期化装置
- アクセス制御システム役割とプロファイルの初期化
- ユーザー・プロファイルにある有効期限データの変更
- ユーザー・プロファイルにあるログオン失敗カウントのリセット
- 共用アクセス制御情報の読み取り
- ユーザー・プロファイルの削除
- 役割の削除

最初は、許可される機能がアクセス制御の初期化に関連する機能となるようにデフォルトの役割が定義されます。したがって、何らかの有益な暗号作業を行う前に、暗号化コプロセッサが必ず初期化されます。この要件により、システムを使用可能状態にするときに誰かが誤って権限を変更し忘れるようなセキュリティ上の「事故」を防ぐことができます。

注：法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

関連概念

18 ページの『セキュア・アクセス』

アクセス制御は、システム資源の使用を、資源との対話を許可されているユーザーのみに制限します。サーバーでは、システム資源へのユーザーの権限を管理することができます。

74 ページの『機能制御ベクトルのロード』

機能制御ベクトルは、鍵を作成するために使用する鍵長を暗号化コプロセッサに通知します。機能制御ベクトルをロードしないと、すべての暗号機能を実行することができません。

役割を定義する：最も簡単、かつ迅速に新しい役割を定義する（およびデフォルトの役割を再定義する）には、暗号化コプロセッサ構成のための Web ベースのユーティリティを使用します。このユーティリティには、`http://server-name:2001` の「System Task」ページからアクセスできます。このユーティリティには、コプロセッサが初期化されていない状態の場合に使用される、基本構成ウィザードが含まれています。基本構成ウィザードは、1 つまたは 3 つの管理役割の定義と、デフォルトの役割の再定義を行うことができます。コプロセッサが初期設定済みである場合は、「構成の管理」、さらに「役割 (Roles)」とクリックして新しい役割を定義するか、既存の役割を変更または削除します。

独自のアプリケーションを作成して、役割を管理することもできます。これを行うには、`Access_Control_Initialization (CSUAACI)` および `Access_Control_Maintenance (CSUAACM)` API verb を使用します。コプロセッサでデフォルトの役割を変更するには、ASCII で適切なパラメーターにエンコードされた「DEFAULT」を指定します。これには ASCII のスペース文字を 1 つ埋め込まなければなりません。それ以外には、役割 ID またはプロファイル ID に使用できる文字には制限がありません。

関連資料

29 ページの『例：コプロセッサ用の役割とプロファイルを作成するための ILE C プログラム』

コプロセッサ用の役割またはプロファイルを作成するには、必要に応じて以下のプログラム例を変更してください。

40 ページの『例：コプロセッサのデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするための ILE C プログラム』

コプロセッサ用のデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするには、必要に応じて以下のプログラム例を変更してください。

45 ページの『例：コプロセッサ用の役割またはプロファイルを作成するための ILE RPG プログラム』

コプロセッサ用の役割およびプロファイルを作成するには、必要に応じて以下のプログラム例を変更してください。

54 ページの『例：コプロセッサのデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするための ILE RPG プログラム』

コプロセッサ用のデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするには、必要に応じて以下のプログラム例を変更してください。

プロファイルを定義する：コプロセッサの役割を作成および定義すると、この役割の下で使用するプロファイルを作成することができます。プロファイルを使用すると、ユーザーは、デフォルトの役割では使用できない、コプロセッサ固有の機能にアクセスすることができます。

最も簡単、かつ迅速に新しいプロファイルを定義するには、暗号化コプロセッサ構成のための Web ベースのユーティリティーを使用します。このユーティリティーには、`http://server-name:2001` の「System Tasks」ページからアクセスできます。このユーティリティーには、コプロセッサが初期化されていない状態の場合に使用される、基本構成ウィザードが含まれています。基本構成ウィザードでは、1 つまたは 3 つの管理プロファイルを定義することができます。コプロセッサが初期化済みである場合は、「構成の管理」→「プロファイル」とクリックして、新しいプロファイルを定義するか、既存のプロファイルを変更または削除します。

独自のアプリケーションを作成して、プロファイルを管理することもできます。これを行うには、`Access_Control_Initialization (CSUAACI)` および `Access_Control_Maintenance (CSUAACM)` API verb を使用します。

関連資料

58 ページの『例: コプロセッサ用の既存のプロファイルを変更するための ILE C プログラム』
コプロセッサ用に既存のプロファイルを変更するには、必要に応じて以下のプログラムを変更してください。

60 ページの『例: コプロセッサ用の既存のプロファイルを変更するための ILE RPG プログラム』
コプロセッサ用に既存のプロファイルを変更するには、必要に応じて以下のプログラムを変更してください。

SSL 用コプロセッサ: SSL にコプロセッサを使用する場合は、デフォルトの役割を少なくとも以下のアクセス制御ポイントに対して許可しなければなりません。

- デジタル署名の生成
- デジタル署名の検査
- PKA 鍵の生成
- PKA 複製鍵の生成
- RSA 暗号化クリア・データ
- RSA 暗号解読クリア・データ
- 保管鍵の削除
- 保管鍵のリスト

暗号化コプロセッサ構成ユーティリティーの基本構成ウィザードは、変更しなくてもデフォルトの役割を SSL に使用できるよう、デフォルトの役割を自動的に再定義します。

セキュリティに関する障害を防ぐため、役割とプロファイルのすべてをセットアップした後で、デフォルトの役割に対して、以下のアクセス制御ポイント (暗号化ハードウェア・コマンドとも呼ばれます) を拒否することを検討してください。

注: 通常の操作に必要なアクセス制御ポイントのみを使用可能にしてください。多くても、特に必要とされる機能のみを使用可能にしてください。どのアクセス制御ポイントが必要であるかを判断するには、「CCA Basic Services Guide」を参照してください。各 API に、その API で必要なアクセス制御ポイントがリストされています。特定の API を使用する必要がない場合は、API で必要とされるアクセス制御ポイントを使用不可にすることを検討してください。

- マスター鍵の最初の部分のロード
- マスター鍵パーツの結合
- マスター鍵の設定
- ランダム・マスター鍵の生成

- | • 新規マスター鍵レジスターのクリア
 - | • 旧マスター鍵レジスターのクリア
 - | • CV の変換
 - | • クロックの設定
 - | **重要:** クロックの設定アクセス制御ポイントをデフォルトの役割で使用不可にする場合は、アクセスを使用不可にする前に、クロックが設定されていることを確認してください。クロックは、ユーザーがログオンしようとした際に、コプロセッサによって使用されます。クロックの設定が誤っている場合、ユーザーはログオンできません。
 - | • 再初期化装置
 - | • アクセス制御システムの初期化
 - | • 認証データの変更 (例えば、パスフレーズ)
 - | • パスワード失敗カウントのリセット
 - | • 共用アクセス制御情報の読み取り
 - | • ユーザー・プロファイルの削除
 - | • 役割の削除
 - | • 機能制御ベクトルのロード
 - | • 機能制御ベクトルのクリア
 - | • ユーザー・ログオフの強制
 - | • EID の設定
 - | • マスター鍵複製制御の初期化
 - | • 公開鍵ハッシュの登録
 - | • 複製あり公開鍵の登録
 - | • 公開鍵の登録
 - | • PKA 複製鍵の生成 (SSL のために必要なアクセス制御ポイント)
 - | • 複製情報がパーツ 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15 を取得
 - | • 複製情報がパーツ 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15 を導入
 - | • 保管鍵の削除 (SSL のために必要なアクセス制御ポイント)
 - | • 保管鍵のリスト (SSL のために必要なアクセス制御ポイント)
 - | • マスター鍵の下での暗号化
 - | • データ鍵のエクスポート
 - | • データ鍵のインポート
 - | • マスター鍵への再暗号化
 - | • マスター鍵からの再暗号化
 - | • 最初の鍵パーツのロード
 - | • 鍵パーツの結合
 - | • 鍵パーツの追加
 - | • 鍵パーツの完了
- | 非常に保護機能の高い環境の場合は、アクセス制御システムを初期化した後で、そのアクセス制御システムをロックすることを検討してください。「アクセス制御の初期化」または「役割の削除」のアクセス制御ポイントの使用を許可しているプロファイルを削除することで、アクセス制御システムを変更不可にすること

ができます。これらのアクセス制御ポイントがないと、役割へさらに変更を加えることが不可能になります。「アクセス制御の初期化」または「役割の削除」のアクセス制御ポイントを使用できる権限があれば、DEFAULT 役割を削除することができます。

DEFAULT 役割を削除すると、初期 DEFAULT 役割が自動的に再作成されます。初期 DEFAULT 役割によって、すべての機能をセットアップすることが可能になります。これらのアクセス制御ポイントへのアクセス権を持っているユーザーは、アクセス制御システムの操作によって無制限の権限を持つことになります。コプロセッサーを通常運用する前に、Access_Control_Maintenance (CSUAACM) および Cryptographic_Facility_Query (CSUACFQ) API verb を使用してアクセス制御のセットアップを監査することができます。

何らかの理由で、予期しない状況応答が戻された場合は、コプロセッサーがセキュリティー・ポリシーに合うように再度構成できるまで、コプロセッサーをアプリケーションに使用しないでください。パスフレーズ(パスワード)を変更できる許可が役割に含まれている場合、すべてのプロファイルのパスフレーズを変更することができます。パスフレーズの変更を許可するかどうか、また、もし許可する場合はこの権限をどの役割(複数の場合もある)に持たせるかを検討してください。

ログオンできないという報告がユーザーから寄せられた場合、そのことが、パスフレーズを変更できる許可を持っている人以外の人(または、許可を持っている人に加えて、それ以外の人)に報告されるようにしてください。セキュリティーが重視される操作では、内部の人間による単独の悪質な行為から防御するために、二重制御を必須として役割を定義することを検討してください。例えば、以下のアクセス制御ポイントのグループを、2 つ以上の役割間で分割することを検討してください。1 人のユーザーにマスター鍵グループにあるすべてのコマンドの使用許可は与えないことをお勧めします。許可を与えた場合、セキュリティーのリスクが発生することもあります。

マスター鍵グループは、次のアクセス制御ポイントで構成されています。

- マスター鍵の最初の部分のロード
- マスター鍵パーツの結合
- マスター鍵の設定
- ランダム・マスター鍵の生成
- 新規マスター鍵レジスターのクリア
- 旧マスター鍵レジスターのクリア

同じ理由から、1 人のユーザーに複製鍵グループ内にあるすべてのコマンドは許可しないでください。

複製鍵グループは、次のアクセス制御ポイントで構成されています。

- マスター鍵複製制御の初期化
- 公開鍵ハッシュの登録
- 複製あり公開鍵の登録
- 公開鍵の登録
- PKA 複製鍵の生成
- 複製情報がパーツ 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15 を取得
- 複製情報がパーツ 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15 を導入

コプロセッサー用のプロファイルを作成、定義した後で、コプロセッサー用の機能制御ベクトルをロードする必要があります。機能制御ベクトルがないと、コプロセッサーはすべての暗号機能を実行することができません。

例: コプロセッサ用の役割とプロファイルを作成するための ILE C プログラム:

コプロセッサ用の役割またはプロファイルを作成するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
/*-----*/
/* CRTROLEPRF */
/* */
/* Sample program to create roles and profiles in the */
/* cryptographic adapter. */
/* */
/* */
/* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 1999 */
/* */
/* This material contains programming source code for your */
/* consideration. These examples have not been thoroughly */
/* tested under all conditions. IBM, therefore, cannot */
/* guarantee or imply reliability, serviceability, or function */
/* of these program. All programs contained herein are */
/* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
/* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
/* these programs and files. */
/* */
/* */
/* Note: Input format is more fully described in Chapter 2 of */
/* IBM CCA Basic Services Reference and Guide */
/* (SC31-8609) publication. */
/* */
/* Parameters: */
/* none. */
/* */
/* Example: */
/* CALL PGM(CRTROLEPRF) */
/* */
/* Use these commands to compile this program on the system: */
/* CRTCMOD MODULE(CRTROLEPRF) SRCFILE(SAMPLE) */
/* CRTPGM PGM(CRTROLEPRF) MODULE(CRTROLEPRF) */
/* BNDSRVPGM(QCCA/CSUAACI QCCA/CSNBOWH) */
/* */
/* Note: Authority to the CSUAACI and CSNBOWH service programs */
/* in the QCCA library is assumed. */
/* */
/* The Common Cryptographic Architecture (CCA) verbs used are */
/* Access_Control_Initialization (CSUAACI) and */
/* One_Way_Hash (CSNBOWH). */
/* */
/* Note: This program assumes the device you want to use is */
/* already identified either by defaulting to the CRP01 */
/* device or has been explicitly named using the */
/* Cryptographic_Resource_Allocate verb. Also this */
/* device must be varied on and you must be authorized */
/* to use this device description. */
/* */
/* Note: Before running this program, the clock in the must be */
/* set using Cryptographic_Facility_Control (CSUACFC) in order */
/* to be able to logon afterwards. */
/* */
```

```

| /*-----*/
| #include "csucincl.h"      /* header file for CCA Cryptographic
|                            Service Provider          */
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
|
| void main(int argc, char *argv[]) {
|
| /*-----*/
| /* standard return codes          */
| /*-----*/
|
| #define ERROR   -1
| #define OK      0
| #define WARNING 4
|
| /*-----*/
| /* Variables used for parameters on CCA APIs          */
| /*-----*/
|
| long return_code;
| long reason_code;
| long exit_data_length;
| char exit_data[2];
| char rule_array[4][8];
| long rule_array_count;
| long verb_data1_length;
| long verb_data2_length;
| long hash_length;
| long text_length;
| char *text;
| char chaining_vector[128];
| long chaining_vector_length;
|
| /*-----*/
| /* Definitions for profiles          */
| /*-----*/
| typedef struct
| {
|     char    version[2];          /* Profile structure version */
|     short   length;              /* length of structure      */
|     char    comment[20];        /* Description                */
|     short   checksum;
|     char    logon_failure_count;
|     char    reserved;
|     char    userid[8];          /* Name for this profile    */
|     char    role[8];            /* Role that profile uses    */
|     short   act_year;           /* Activation date - year    */
|     char    act_month;          /* Activation date - month   */
|     char    act_day;            /* Activation date - day     */
|     short   exp_year;           /* Expiration date - year   */
|     char    exp_month;          /* Expiration date - month  */
|     char    exp_day;            /* Expiration date - day    */
|     short   total_auth_data_length;
|     short   field_type;
|     short   auth_data_length_1;
|     short   mechanism;          /* Authentication mechanism */
|     short   strength;           /* Strength of mechanism    */
|     short   mech_exp_year;      /* Mechanism expiration - year*/
|     char    mech_exp_month;     /* Mech. expiration - month */
|     char    mech_exp_day;       /* Mechansim expiration - day */
|     char    attributes[4];
|     char    auth_data[20];      /* Secret data              */
| } profile_T;
|
| typedef struct

```

```

|     {
|         long         number;           /* Number profiles in struct */
|         long         reserved;
|         profile_T    profile[3];
|     } aggregate_profile;
|
| aggregate_profile * verb_data1;       /* Aggregate structure for    */
|                                       /* defining profiles          */
|
| /*-----*/
| /* Definitions for roles              */
| /*-----*/
| /*-----*/
| /* Default role - access control points list - */
| /*         authorized to everything EXCEPT:   */
| /* 0x0018 - Load 1st part of Master Key       */
| /* 0x0019 - Combine Master Key Parts          */
| /* 0x001A - Set Master Key                   */
| /* 0x0020 - Generate Random Master Key       */
| /* 0x0032 - Clear New Master Key Register    */
| /* 0x0033 - Clear Old Master Key Register    */
| /* 0x0053 - Load 1st part of PKA Master Key  */
| /* 0x0054 - Combine PKA Master Key Parts     */
| /* 0x0057 - Set PKA Master Key              */
| /* 0x0060 - Clear New PKA Master Key Register */
| /* 0x0061 - Clear Old PKA Master Key Register */
| /* 0x0110 - Set Clock                       */
| /* 0x0111 - Reinitialize device             */
| /* 0x0112 - Initialize access control system */
| /* 0x0113 - Change user profile expiration date */
| /* 0x0114 - Change authentication data (eg. passphrase) */
| /* 0x0115 - Reset password failure count    */
| /* 0x0116 - Read Public Access Control Information */
| /* 0x0117 - Delete user profile            */
| /* 0x0118 - Delete role                   */
| /* 0x0119 - Load Function Control Vector    */
| /* 0x011A - Clear Function Control Vector   */
| /* 0x011B - Force User Logoff              */
| /* 0x0200 - Register PKA Public Key Hash    */
| /* 0x0201 - Register PKA Public Key, with cloning */
| /* 0x0202 - Register PKA Public Key       */
| /* 0x0203 - Delete Retained Key           */
| /* 0x0204 - PKA Clone Key Generate        */
| /* 0x0211 - 0x21F - Clone information - obtain 1-15 */
| /*-----*/
| /* For access control points 0x01 - 0x127 */
| char default_bitmap[] =
|     { 0x00, 0x03, 0xF0, 0x1D, 0x00, 0x00, 0x00, 0x00,
|       0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
|       0x00, 0x0A, 0x80, 0x00, 0x88, 0x2F, 0x71, 0x10,
|       0x10, 0x04, 0x03, 0x31, 0x80, 0x00, 0x00, 0x00,
|       0xFF, 0x7F, 0x40, 0x6B, 0x80};
|
| /* For access control points 0x200 - 0x23F */
| char default2_bitmap[] =
|     { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE6, 0x0F };
|
| /*-----*/
| /* role #1 - authorized to same as default plus also */
| /*         authorized to:                             */
| /* 0x0018 - Load 1st part of Master Key       */
| /* 0x0020 - Generate Random Master Key       */
| /* 0x0032 - Clear New Master Key Register    */
| /* 0x0053 - Load 1st part of PKA Master Key  */
| /* 0x0060 - Clear New PKA Master Key Register */
| /* 0x0119 - Load Function Control Vector    */
| /* 0x0201 - Register PKA Public Key, with cloning */

```

```

| /* 0x0202 - Register PKA Public Key */
| /* 0x0203 - Delete Retained Key */
| /* 0x0204 - PKA Clone Key Generate */
| /* 0x0211 - 0x215 - Clone information - obtain 1-5 */
| /* 0x0221 - 0x225 - Clone information - install 1-5 */
| /*-----*/
| char role1_bitmap[] =
| { 0x00, 0x03, 0xF0, 0x9D, 0x80, 0x00, 0x20, 0x00,
|   0x80, 0x00, 0x10, 0x00, 0x80, 0x00, 0x00, 0x00,
|   0x00, 0x0A, 0x80, 0x00, 0x88, 0x1F, 0x71, 0x10,
|   0x10, 0x04, 0x03, 0x11, 0x80, 0x00, 0x00, 0x00,
|   0xFF, 0x7F, 0x00, 0x4F, 0x80};
| char role1_bitmap2[] =
| { 0x78, 0x00, 0x7C, 0x00, 0x7C, 0x00, 0xE6, 0x0F };
|
| /*-----*/
| /* role #2 - authorized to same as default plus also */
| /*     authorized to: */
| /* 0x0019 - Combine Master Key Parts */
| /* 0x001A - Set Master Key */
| /* 0x0033 - Clear Old Master Key Register */
| /* 0x0054 - Combine PKA Master Key Parts */
| /* 0x0057 - Set PKA Master Key */
| /* 0x0061 - Clear Old Master Key Register */
| /* 0x011A - Clear Function Control Vector */
| /* 0x0200 - Register PKA Public Key Hash */
| /* 0x0201 - Register PKA Public Key, with cloning */
| /* 0x0203 - Delete Retained Key */
| /* 0x0204 - PKA Clone Key Generate */
| /* 0x0216 - 0x21A - Clone information - obtain 6-10 */
| /* 0x0226 - 0x22A - Clone information - install 6-10 */
| /*-----*/
| char role2_bitmap[] =
| { 0x00, 0x03, 0xF0, 0x7D, 0x80, 0x00, 0x10, 0x00,
|   0x80, 0x00, 0x09, 0x00, 0x40, 0x00, 0x00, 0x00,
|   0x00, 0x0A, 0x80, 0x00, 0x88, 0x1F, 0x71, 0x10,
|   0x10, 0x04, 0x03, 0x31, 0x80, 0x00, 0x00, 0x00,
|   0xFF, 0x7F, 0x00, 0x2F, 0x80};
| char role2_bitmap2[] =
| { 0xD8, 0x00, 0x03, 0xE0, 0x03, 0xE0, 0xE6, 0x0F };
|
| /*-----*/
| /* role #3 - authorized to same as default plus also */
| /*     authorized to: */
| /* 0x0110 - Set Clock */
| /* 0x0111 - Reinitialize device */
| /* 0x0112 - Initialize access control system */
| /* 0x0113 - Change user profile expiration date */
| /* 0x0114 - Change authentication data (eg. passphrase) */
| /* 0x0115 - Reset password failure count */
| /* 0x0116 - Read Public Access Control Information */
| /* 0x0117 - Delete user profile */
| /* 0x0118 - Delete role */
| /* 0x011B - Force User Logoff */
| /* 0x0200 - Register PKA Public Key Hash */
| /* 0x0201 - Register PKA Public Key, with cloning */
| /* 0x0203 - Delete Retained Key */
| /* 0x0204 - PKA Clone Key Generate */
| /* 0x021B - 0x21F - Clone information - obtain 11-15 */
| /* 0x022B - 0x22F - Clone information - install 11-15 */
| /*-----*/
| char role3_bitmap[] =
| { 0x00, 0x03, 0xF0, 0x1D, 0x00, 0x00, 0x00, 0x00,
|   0x80, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00,
|   0x00, 0x0A, 0x80, 0x00, 0x88, 0x1F, 0x71, 0x10,
|   0x10, 0x04, 0x03, 0x31, 0x80, 0x00, 0x00, 0x00,
|   0xFF, 0x7F, 0xFF, 0x9F, 0x80};

```



```

                * 6 + /* 3 roles * 2 segments each */
                sizeof(default_bitmap) * 3 +
                sizeof(default2_bitmap) * 3);

work_ptr = verb_data2;          /* Set working pointer to
                                start of verb data 2 storage */

aggregate_role_header.number = 3; /* Define/replace 3 roles */
aggregate_role_header.reserved = 0;
                                /* Copy header into verb data
                                2 storage. */
memcpy(work_ptr, (void*)&aggregate_role_header,
        sizeof(aggregate_role_header));

                                /* Adjust work pointer to point
                                after header. */
work_ptr += sizeof(aggregate_role_header);

/*-----*/
/* Fill in the fields of the role definitions. */
/* Each role is version 1, has authentication strength of 0, */
/* has valid time from 12:00 Midnight (0) to 23:59 (x173B), */
/* is valid every day of the week. (xFE is 7 bits set), */
/* has one access control points segment that starts at bit 0 */
/* and goes to bit x11F, and has 20 spaces for a comment. */
/*-----*/
role_header.version[0]          = 1;
role_header.version[1]          = 0;
role_header.length              = sizeof(role_header) +
                                sizeof(access_control_points_header) +
                                2 * sizeof(access_control_points_segment_header) +
                                sizeof(default_bitmap) + sizeof(default2_bitmap);
role_header.checksum            = 0;
role_header.reserved1           = 0;
role_header.auth_strength       = 0;
role_header.lower_time          = 0;
role_header.upper_time          = 0x173B;
role_header.valid_days_of_week = 0xFE;
role_header.reserved2           = 0;
memset(role_header.comment, ' ', 20);

access_control_points_header.number_segments = 2;
access_control_points_header.reserved       = 0;
access_control_points_segment_header.reserved = 0;

for (i=0; i<3; i++)
{
    switch (i) {
        /*-----*/
        /* Set name for ROLE1 */
        /*-----*/
        case 0:
            memcpy(role_header.role, "ROLE1 ", 8);
            bitmap1 = role1_bitmap;
            bitmap2 = role1_bitmap2;

            break;

            /*-----*/
            /* Set name for ROLE2 */
            /*-----*/
        case 1:
            memcpy(role_header.role, "ROLE2 ", 8);
            bitmap1 = role2_bitmap;
            bitmap2 = role2_bitmap2;
            break;
    }
}

```



```

        /*-----*/
        /* Set name for ROLE3 */
        /*-----*/
    case 2:
        memcpy(role_header.role, "ROLE3 ", 8);
        bitmap1 = role3_bitmap;
        bitmap2 = role3_bitmap2;
    }

    /*-----*/
    /* Copy role header */
    /*-----*/
    memcpy(work_ptr, (void*)&role_header, sizeof(role_header));

        /* Adjust work pointer to
        point after role header. */
    work_ptr += sizeof(role_header);

    /*-----*/
    /* Copy access control points header */
    /*-----*/
    memcpy(work_ptr,
        (void *)&access_control_points_header,
        sizeof(access_control_points_header));

        /* Adjust work pointer to
        point after header. */
    work_ptr += sizeof(access_control_points_header);

    /*-----*/
    /* Copy access control points segment 1 */
    /*-----*/
    access_control_points_segment_header.start_bit = 0;
    access_control_points_segment_header.end_bit = 0x127;
    access_control_points_segment_header.number_bytes =
        sizeof(default_bitmap);
    memcpy(work_ptr,
        (void *)&access_control_points_segment_header,
        sizeof(access_control_points_segment_header));

        /* Adjust work pointer to
        point after header. */
    work_ptr += sizeof(access_control_points_segment_header);

    /*-----*/
    /* Copy access control points segment 1 bitmap */
    /*-----*/
    memcpy(work_ptr, bitmap1, sizeof(default_bitmap));

        /* Adjust work pointer to
        point after bitmap. */
    work_ptr += sizeof(default_bitmap);

    /*-----*/
    /* Copy access control points segment 2 */
    /*-----*/
    access_control_points_segment_header.start_bit = 0x200;
    access_control_points_segment_header.end_bit = 0x23F;
    access_control_points_segment_header.number_bytes =
        sizeof(default2_bitmap);

    memcpy(work_ptr,
        (void *)&access_control_points_segment_header,
        sizeof(access_control_points_segment_header));

        /* Adjust work pointer to
        point after header. */

```

```

work_ptr += sizeof(access_control_points_segment_header);

/*-----*/
/* Copy access control points segment 2 bitmap      */
/*-----*/
memcpy(work_ptr, bitmap2, sizeof(default2_bitmap));

/* Adjust work pointer to
   point after bitmap. */
work_ptr += sizeof(default2_bitmap);
}

/*-----*/
/* Allocate storage for aggregate profile structure */
/*-----*/
verb_data1 = malloc(sizeof(aggregate_profile));

verb_data1->number = 3; /* Define 3 profiles */
verb_data1->reserved = 0;

/*-----*/
/* Each profile: */
/* will be version 1, */
/* have an activation date of 1/1/00, */
/* have an expiration date of 6/30/2005, */
/* use passphrase hashed with SHA1 for the mechanism (0x0001), */
/* will be renewable (attributes = 0x8000) */
/* and has 20 spaces for a comment */
/*-----*/
for (i=0; i<3; i++)
{
    verb_data1->profile[i].length = sizeof(profile_T);
    verb_data1->profile[i].version[0] = 1;
    verb_data1->profile[i].version[1] = 0;
    verb_data1->profile[i].checksum = 0;
    verb_data1->profile[i].logon_failure_count = 0;
    verb_data1->profile[i].reserved = 0;
    verb_data1->profile[i].act_year = 2000;
    verb_data1->profile[i].act_month = 1;
    verb_data1->profile[i].act_day = 1;
    verb_data1->profile[i].exp_year = 2005;
    verb_data1->profile[i].exp_month = 6;
    verb_data1->profile[i].exp_day = 30;
    verb_data1->profile[i].total_auth_data_length = 0x24;
    verb_data1->profile[i].field_type = 0x0001;
    verb_data1->profile[i].auth_data_length_1 = 0x20;
    verb_data1->profile[i].mechanism = 0x0001;
    verb_data1->profile[i].strength = 0;
    verb_data1->profile[i].mech_exp_year = 2005;
    verb_data1->profile[i].mech_exp_month = 6;
    verb_data1->profile[i].mech_exp_day = 30;
    verb_data1->profile[i].attributes[0] = 0x80;
    verb_data1->profile[i].attributes[1] = 0;
    verb_data1->profile[i].attributes[2] = 0;
    verb_data1->profile[i].attributes[3] = 0;

    memset(verb_data1->profile[i].comment, ' ', 20);

    memcpy(rule_array, "SHA-1 ", 8);
    rule_array_count = 1;
    chaining_vector_length = 128;
    hash_length = 20;

    switch (i) {
        /*-----*/
        /* Set name, role, passphrase of profile 1 */

```

```

/*-----*/
case 0:
    memcpy(verb_data1->profile[i].userid,"SECOFR1 ",8);
    memcpy(verb_data1->profile[i].role, "ROLE1  ",8);
    text_length = 10;
    text        = "Is it safe";
    break;
/*-----*/
/* Set name, role, passphrase of profile 2 */
/*-----*/
case 1:
    memcpy(verb_data1->profile[i].userid,"SECOFR2 ",8);
    memcpy(verb_data1->profile[i].role, "ROLE2  ",8);
    text_length = 18;
    text        = "I think it is safe";
    break;
/*-----*/
/* Set name, role, passphrase of profile 3 */
/*-----*/
case 2:
    memcpy(verb_data1->profile[i].userid,"SECOFR3 ",8);
    memcpy(verb_data1->profile[i].role, "ROLE3  ",8);
    text_length = 12;
    text        = "Is what safe";
}

/*-----*/
/* Call One_Way_Hash to hash the pass-phrase */
/*-----*/
CSNBOWH( &return_code,
         &reason_code,
         &exit_data_length,
         exit_data,
         &rule_array_count,
         (char*)rule_array,
         &text_length,
         text,
         &chaining_vector_length,
         chaining_vector,
         &hash_length,
         verb_data1->profile[i].auth_data);
}

/*-----*/
/* Call Access_Control_Initialize (CSUAACI) to create */
/* the roles and profiles. */
/*-----*/
rule_array_count = 2;
memcpy(rule_array, "INIT-AC REPLACE ", 16);
verb_data1_length = sizeof(aggregate_profile);
verb_data2_length = sizeof(aggregate_role_header) +
                    sizeof(role_header) * 3 +
                    sizeof(access_control_points_header) * 3 +
                    sizeof(access_control_points_segment_header)
                    * 6 + /* 3 roles * 2 segments each */
                    sizeof(default_bitmap) * 3 +
                    sizeof(default2_bitmap) * 3;

CSUAACI( &return_code,
         &reason_code,
         &exit_data_length,
         exit_data,
         &rule_array_count,
         (char *)rule_array,
         (long *) &verb_data1_length,
         (char *) verb_data1,
         (long *) &verb_data2_length,

```

```

        (char *) verb_data2);

if (return_code > WARNING)
    printf("Access_Control_Initialize failed. Return/reason codes ¥
%d/%d¥n",return_code, reason_code);
else
    printf("The new roles and profiles were successfully created¥n");

/*-----*/
/* The Access_Control_Initialize SAPI verb needs to be      */
/* called one more time to replace the DEFAULT role so that */
/* a user that does not log on is not able to change any    */
/* settings in the .                                         */
/*-----*/
work_ptr = verb_data2;          /* Set working pointer to
                                start of verb data 2 storage */

aggregate_role_header.number = 1; /* Define/replace 1 roles */
aggregate_role_header.reserved = 0;
memcpy(work_ptr,(void*)&aggregate_role_header,
        sizeof(aggregate_role_header));

                                /* Adjust work pointer to
                                point after header. */
work_ptr += sizeof(aggregate_role_header);

/*-----*/
/* Fill in the fields of the role definitions.              */
/* Each role is version 1, has authentication strength of 0, */
/* has valid time from 12:00 Midnight (0) to 23:59 (x173B), */
/* is valid every day of the week. (xFE is 7 bits set),     */
/* has one access control points segment that starts at bit 0 */
/* and goes to bit x11F, and has 20 spaces for a comment.   */
/*-----*/
role_header.version[0]          = 1;
role_header.version[1]          = 0;
role_header.length              = sizeof(role_header) +
                                sizeof(access_control_points_header) +
                                2 * sizeof(access_control_points_segment_header) +
                                sizeof(default_bitmap) + sizeof(default2_bitmap);
role_header.checksum            = 0;
role_header.reserved1           = 0;
role_header.auth_strength       = 0;
role_header.lower_time          = 0;
role_header.upper_time          = 0x173B;
role_header.valid_days_of_week  = 0xFE;
role_header.reserved2           = 0;
memset(role_header.comment,' ', 20);

access_control_points_header.number_segments = 2;
access_control_points_header.reserved       = 0;
access_control_points_segment_header.reserved = 0;

                                /* DEFAULT role id must be in */
                                /* ASCII representation.      */
memcpy(role_header.role, "%x44¥x45¥x46¥x41¥x55¥x4C¥x54¥x20", 8);
bitmap1 = default_bitmap;
bitmap2 = default2_bitmap;

/*-----*/
/* Copy role header                                          */
/*-----*/
memcpy(work_ptr,(void*)&role_header, sizeof(role_header));

                                /* Adjust work pointer to
                                point after header. */
work_ptr += sizeof(role_header);

```

```

/*-----*/
/* Copy access control points header */
/*-----*/
memcpy(work_ptr,
        (void *)&access_control_points_header,
        sizeof(access_control_points_header));

        /* Adjust work pointer to
        point after header. */
work_ptr += sizeof(access_control_points_header);

/*-----*/
/* Copy access control points segment 1 */
/*-----*/
access_control_points_segment_header.start_bit = 0;
access_control_points_segment_header.end_bit = 0x127;
access_control_points_segment_header.number_bytes =
        sizeof(default_bitmap);
memcpy(work_ptr,
        (void *)&access_control_points_segment_header,
        sizeof(access_control_points_segment_header));

        /* Adjust work pointer to
        point after header. */
work_ptr += sizeof(access_control_points_segment_header);

/*-----*/
/* Copy access control points segment 1 bitmap */
/*-----*/
memcpy(work_ptr, bitmap1, sizeof(default_bitmap));

        /* Adjust work pointer to
        point after bitmap. */
work_ptr += sizeof(default_bitmap);

/*-----*/
/* Copy access control points segment 2 */
/*-----*/
access_control_points_segment_header.start_bit = 0x200;
access_control_points_segment_header.end_bit = 0x23F;
access_control_points_segment_header.number_bytes =
        sizeof(default2_bitmap);

memcpy(work_ptr,
        (void *)&access_control_points_segment_header,
        sizeof(access_control_points_segment_header));

        /* Adjust work pointer to
        point after header. */
work_ptr += sizeof(access_control_points_segment_header);

/*-----*/
/* Copy access control points segment 2 bitmap */
/*-----*/
memcpy(work_ptr, bitmap2, sizeof(default2_bitmap));

rule_array_count = 2;
memcpy(rule_array, "INIT-AC REPLACE ", 16);
verb_data1_length = 0;
verb_data2_length = sizeof(aggregate_role_header) +
        sizeof(role_header) +
        sizeof(access_control_points_header) +
        sizeof(access_control_points_segment_header)
        * 2 +
        sizeof(default_bitmap) +
        sizeof(default2_bitmap);

```

```

| CSUAACI( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (char *)rule_array,
|         (long *) &verb_data1_length,
|         (char *) verb_data1,
|         (long *) &verb_data2_length,
|         (char *) verb_data2);
|
| if (return_code > 4)
|   printf("The default role was not replaced. Return/reason code:¥
|         %d/¥d¥n",return_code, reason_code);
| else
|   printf("The default role was successfully updated.¥n");
| }

```

関連概念

25 ページの『役割を定義する』

例: コプロセッサのデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするための ILE C プログラム:

コプロセッサ用のデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| /*-----*/
| /* SETDEFAULT */
| /* */
| /* Sample program to authorize the default role to all access */
| /* control points in the . */
| /* */
| /* */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000 */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* none. */
| /* */
| /* Example: */
| /* CALL PGM(SETDEFAULT) */

```

```

| /* */
| /* Use these commands to compile this program on the system: */
| /* CRTCMOD MODULE(SETDEFAULT) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(SETDEFAULT) MODULE(SETDEFAULT) */
| /*     BNDSRVPGM(QCCA/CSUAACI) */
| /* */
| /* Note: Authority to the CSUAACI service programs */
| /*     in the QCCA library is assumed. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Access_Control_Initialization (CSUAACI). */
| /* */
| /* Note: This program assumes the device you want to use is */
| /*     already identified either by defaulting to the CRP01 */
| /*     device or has been explicitly named using the */
| /*     Cryptographic_Resource_Allocate verb. Also this */
| /*     device must be varied on and you must be authorized */
| /*     to use this device description. */
| /* */
| /*-----*/
|
| #include "csucincl.h" /* header file for CCA Cryptographic
|                       Service Provider */
|
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
|
| void main(int argc, char *argv[]) {
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR    -1
| #define OK       0
| #define WARNING  4
|
| /*-----*/
| /* parameters for CCA APIs */
| /*-----*/
|
|     long return_code;
|     long reason_code;
|     long exit_data_length;
|     char exit_data[2];
|     char rule_array[4][8];
|     long rule_array_count;
|     long verb_data1_length;
|     long verb_data2_length;
|     char  verb_data1[4];
|
| /*-----*/
| /* Structure for defining a role */
| /*-----*/
| struct role_header
| {
|     char          version[2];
|     short         length;
|     char          comment[20];
|     short         checksum;
|     short         reserved1;
|     char          role[8];
|     short         auth_strength;
|     char          lower_time_hour;
|     char          lower_time_minute;
|     char          upper_time_hour;
|     char          upper_time_minute;

```

```

char          valid_days_of_week;
char          reserved2;
} role_header;

/*-----*/
/* Structure for defining aggregate roles */
/*-----*/
struct aggregate_role
{
    long      number;
    long      reserved;
} aggregate_role_header;

/*-----*/
/* Structures for defining the access control points in a role */
/*-----*/
struct access_control_points_header
{
    short     number_segments;    /* Number of segments of */
                                   /* the access points map */
    short     reserved;
} access_control_points_header;

struct access_control_points_segment_header
{
    short     start_bit;          /* Starting bit in this */
                                   /* segment. */
    short     end_bit;            /* Ending bit */
    short     number_bytes;       /* Number of bytes in */
                                   /* this segment */
    short     reserved;
} access_control_points_segment_header;

/*-----*/
/* Default role - access control points list - */
/*          authorized to everything */
/*          */
/* For access control points 0x01 - 0x12F */
/*-----*/
char default_bitmap[] =
{ 0x00, 0x03, 0xF0, 0xFD, 0x80, 0x00, 0x30, 0x00,
  0x80, 0x00, 0x19, 0x00, 0xC0, 0x00, 0x00, 0x00,
  0x00, 0x0A, 0x80, 0x00, 0x88, 0x2F, 0x71, 0x10,
  0x18, 0x04, 0x03, 0x31, 0x80, 0x00, 0x00, 0x00,
  0xFF, 0x7F, 0xFF, 0xFF, 0x80};

/*-----*/
/* For access control points 0x200 - 0x23F */
/*-----*/
char default2_bitmap[] =
{ 0xF8, 0x00, 0x7F, 0xFF, 0x7F, 0xFF, 0xE6, 0x0F };

unsigned char * verb_data2;
unsigned char * work_ptr;

int i;          /* Loop counter */

/*-----*/
/* Start of code */
/*-----*/

/*-----*/
/* Allocate storage for the aggregate role structure */
/*-----*/
verb_data2 = malloc(sizeof(aggregate_role_header) +
                    sizeof(role_header) +

```



```

        sizeof(access_control_points_header) +
        sizeof(access_control_points_segment_header)
        * 2 +
        sizeof(default_bitmap) +
        sizeof(default2_bitmap));

work_ptr = verb_data2;          /* Set up work pointer */

aggregate_role_header.number = 1; /* Define/replace 1 role */
aggregate_role_header.reserved = 0; /* Initialize reserved field*/

                                /* Copy header to verb_data2
                                storage. */
memcpy(work_ptr, (void*)&aggregate_role_header,
        sizeof(aggregate_role_header));

work_ptr += sizeof(aggregate_role_header); /* Set work pointer
after role header */

/*-----*/
/* Fill in the fields of the role definition. */
/*-----*/
role_header.version[0] = 1; /* Version 1 role */
role_header.version[1] = 0;

                                /* Set length of the role */
role_header.length = sizeof(role_header)
                    + sizeof(access_control_points_header)
                    + 2 *
                    sizeof(access_control_points_segment_header)
                    + sizeof(default_bitmap)
                    + sizeof(default2_bitmap);

role_header.checksum = 0; /* Checksum is not used */
role_header.reserved1 = 0; /* Reserved must be 0 */
role_header.auth_strength = 0; /* Authentication strength */
                                /* is set to 0. */

                                /* Lower time is 00:00 */
role_header.lower_time_hour = 0;
role_header.lower_time_minute = 0;

                                /* Upper time is 23:59 */
role_header.upper_time_hour = 23;
role_header.upper_time_minute = 59;
role_header.valid_days_of_week = 0xFE; /* Valid every day */
                                /* 7 bits - 1 bit each day */

role_header.reserved2 = 0; /* Reserved must be 0 */

                                /* Role is DEFAULT */
                                /* expressed in ASCII */
memcpy(role_header.role, "%x44%x45%x46%x41%x55%x4C%x54%x20", 8);

memset(role_header.comment, ' ', 20); /* No description for role */

/*-----*/
/* Copy role header into verb_data2 storage */
/*-----*/
memcpy(work_ptr, (void*)&role_header, sizeof(role_header));
work_ptr += sizeof(role_header);

/*-----*/
/* Set up access control points header and then */
/* copy it into verb_data2 storage. */
/*-----*/
access_control_points_header.number_segments = 2;
access_control_points_header.reserved = 0;

```

```

| access_control_points_segment_header.reserved      = 0;
|
| memcpy(work_ptr,
|         (void *)&access_control_points_header,
|         sizeof(access_control_points_header));
|
|         /* Adjust work_ptr to point to the
|         first segment          */
| work_ptr += sizeof(access_control_points_header);
|
| /*-----*/
| /* Set up the segment header for segment 1 and then */
| /* copy into verb_data2 storage          */
| /*-----*/
| access_control_points_segment_header.start_bit = 0;
| access_control_points_segment_header.end_bit  = 0x127;
| access_control_points_segment_header.number_bytes =
|         sizeof(default_bitmap);
| memcpy(work_ptr,
|         (void *)&access_control_points_segment_header,
|         sizeof(access_control_points_segment_header));
|
|         /* Adjust work_ptr to point to the
|         first segment bitmap      */
| work_ptr += sizeof(access_control_points_segment_header);
|
| /*-----*/
| /* Copy access control points segment 1 bitmap      */
| /*-----*/
| memcpy(work_ptr, default_bitmap, sizeof(default_bitmap));
|
|         /* Adjust work_ptr to point to the
|         second segment          */
| work_ptr += sizeof(default_bitmap);
|
| /*-----*/
| /* Set up the segment header for segment 2 and then */
| /* copy into verb_data2 storage          */
| /*-----*/
| access_control_points_segment_header.start_bit = 0x200;
| access_control_points_segment_header.end_bit  = 0x23F;
| access_control_points_segment_header.number_bytes =
|         sizeof(default2_bitmap);
|
| memcpy(work_ptr,
|         (void *)&access_control_points_segment_header,
|         sizeof(access_control_points_segment_header));
|
|         /* Adjust work_ptr to point to the
|         second segment bitmap      */
| work_ptr += sizeof(access_control_points_segment_header);
|
| /*-----*/
| /* Copy access control points segment 2 bitmap      */
| /*-----*/
| memcpy(work_ptr, default2_bitmap, sizeof(default2_bitmap));
|
| /*-----*/
| /* Set the length of verb data 2 (Role definition) */
| /*-----*/
| verb_data2_length = sizeof(aggregate_role_header) +
|         role_header.length;
|
| /*-----*/
| /* Set remaining parameters          */
| /*-----*/
| rule_array_count = 2;

```

```

| memcpy(rule_array, "INIT-AC REPLACE ", 16);
| verb_data1_length = 0;
|
| /*-----*/
| /* Call Access_Control_Initialize (CSUAACI) to set the */
| /* default role. */
| /*-----*/
| CSUAACI( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char *)rule_array,
|         &verb_data1_length,
|         (unsigned char *) verb_data1,
|         &verb_data2_length,
|         verb_data2);
|
| if (return_code > 4)
|     printf("The default role was not replaced. Return/reason code:¥
|           %d/¥d¥n",return_code, reason_code);
| else
|     printf("The default role was successfully updated.¥n");
| }

```

関連概念

25 ページの『役割を定義する』

例: コプロセッサ用の役割またはプロファイルを作成するための ILE RPG プログラム:

コプロセッサ用の役割およびプロファイルを作成するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| D*****
| D* CRTROLEPRF
| D*
| D* Sample program to create 3 roles and 3 profiles in the
| D* and change the authority for the default role.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D*       IBM CCA Basic Services Reference and Guide
| D*       (SC31-8609) publication.
| D*

```

```

| D* Parameters: None
| D*
| D* Example:
| D* CALL PGM(CRTROLEPRF)
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(CRTROLEPRF) SRCFILE(SAMPLE)
| D* CRTPGM PGM(CRTROLEPRF) MODULE(CRTROLEPRF)
| D* BNDDIR(QCCA/QC6BNDDIR)
| D*
| D* Note: Authority to the CSUAACI service program in the
| D* QCCA library is assumed.
| D*
| D* The Common Cryptographic Architecture (CCA) verbs used are
| D* Access_Control_Initialize (CSUAACI)
| D*
| D*****
| D*-----
| D* Declare variables used by CCA SAPI calls
| D*-----
| D* ** Return code
| DRETURNCODE S 9B 0
| D* ** Reason code
| DREASONCODE S 9B 0
| D* ** Exit data length
| DEXITDATALEN S 9B 0
| D* ** Exit data
| DEXITDATA S 4
| D* ** Rule array count
| DRULEARRAYCNT S 9B 0
| D* ** Rule array
| DRULEARRAY S 16
| D* ** Text length
| DTEXTLEN S 9B 0
| D* ** Text to hash
| DTEXT S 20
| D* ** Chaining vector length
| DCHAINVCTLEN S 9B 0 INZ(128)
| D* ** Chaining vector
| DCHAINVCT S 128
| D* ** Hash length
| DHASHLEN S 9B 0 INZ(20)
| D*-----
| D* VERBDATA1 contains the aggregate profile structure which
| D* in turn contains 3 profiles.
| D*-----
| DVERBDATALEN1 S 9B 0 INZ(278)
| DVERBDATA1 DS 278
| D* ** Define 3 Profiles
| DNUMPROFS 9B 0 INZ(3)
| D* ** Reserved field
| DRESR1 9B 0 INZ(0)
| DPROF1 90
| DPROF2 90
| DPROF3 90
| D*-----
| D* Define the profile structure
| D*-----
| DPROFILESTRUCT DS
| D* ** Version 1 struct
| DPROFVERS 2 INZ(X'0100')
| D* ** Length of profile
| DPROFLEN 2 INZ(X'005A')
| D* ** Description of profile
| DCOMMENTP 20 INZ(' ')
| D* ** Checksum is not used

```

```

|      DCHECKSUMP          2      INZ(X'0000')
|      D*                  ** Logon failure count
|      DLOGFC              1      INZ(X'00')
|      D*                  ** Reserved
|      DRESR2              1      INZ(X'00')
|      D*                  ** Profile name
|      DUSERID             8
|      D*                  ** Role used
|      DROLENAME           8
|      D*                  ** Activation year (2000)
|      DACTYEAR            2      INZ(X'07D0')
|      D*                  ** Activation month (01)
|      DACTMONTH           1      INZ(X'01')
|      D*                  ** Activation day (01)
|      DACTDAY             1      INZ(X'01')
|      D*                  ** Expiration year (2004)
|      DEXPYEAR            2      INZ(X'07D4')
|      D*                  ** Expiration month (12)
|      DEXPMONTH           1      INZ(X'0C')
|      D*                  ** Expiration day (31)
|      DEXPDAY             1      INZ(X'1F')
|      D*                  ** Total authentication
|      D*                  ** data length
|      DTOTAUTDALEN        2      INZ(X'0024')
|      D*                  ** Field type
|      DFIELDTYPE          2      INZ(X'0001')
|      D*                  ** Authentication data len
|      DAUTDATLEN          2      INZ(X'0020')
|      D*                  ** Authentication mechanism
|      DMECHANISM          2      INZ(X'0001')
|      D*                  ** Mechanism strength
|      DSTRENGTH           2      INZ(X'0000')
|      D*                  ** Mech expiration year (2004)
|      DMCHEXPYEAR         2      INZ(X'07D4')
|      D*                  ** Mech expiration month (12)
|      DMCHEXPMONTH        1      INZ(X'0C')
|      D*                  ** Mech expiration day (31)
|      DMCHEXPDAY          1      INZ(X'1F')
|      D*                  ** Attributes
|      DATTRIBUTES         4      INZ(X'80000000')
|      D*                  ** Authentication data
|      DAUTHDATA           20     INZ('          ')
|      D*
|      D*-----
|      D* The Default role is being replaced
|      D*   Verb_data_2 length set to the length of the default role
|      D*-----
|      DVERBDATALEN2      S          9B 0 INZ(335)
|      D*-----
|      D* VERBDATA2 contains the aggregate role structure which
|      D*   in turn contains 3 roles.
|      D*-----
|      DVERBDATA2        DS
|      D*                  ** Define 3 Roles
|      DNUMROLES          9B 0 INZ(3)
|      D*                  ** Reserved field
|      DRESR3              9B 0 INZ(0)
|      DROLE1              109
|      DROLE2              109
|      DROLE3              109
|      D*
|      D*-----
|      D* Define the role structure
|      D*-----
|      DROLESTRUCT        DS
|      D*                  ** Version 1 struct
|      DROLEVERS          2      INZ(X'0100')

```

```

| D*          ** Length of role
| DROLELEN          2  INZ(X'006D')
| D*          ** Description of role
| DCOMMENTR        20  INZ('          ')
| D*          ** Checksum is not used
| DCHECKSUMR        2  INZ(X'0000')
| D*          ** Reserved field
| DRESR4            2  INZ(X'0000')
| D*          ** Role Name
| DROLE            8
| D*          ** Authentication strength is set to 0
| DAUTHSTRN         2  INZ(X'0000')
| D*          ** Lower time is 00:00
| DLWRTIMHR         1  INZ(X'00')
| DLWRTIMMN         1  INZ(X'00')
| D*          ** Upper time is 23:59
| DUPRTIMHR         1  INZ(X'17')
| DUPRTIMMN         1  INZ(X'3B')
| D*          ** Valid days of week
| DVALIDDOW         1  INZ(X'FE')
| D*          ** Reserved field
| DRESR5            1  INZ(X'00')
| D*          ** 2 Access control points segments are defined
| DNUMSEG           2  INZ(X'0002')
| D*          ** Reserved field
| DRESR6            2  INZ(X'0000')
| D*          ** Starting bit of segment 1 is 0
| DSTART1           2  INZ(X'0000')
| D*          ** Ending bit of segment 1 is 295 (Hex 127).
| DEND1             2  INZ(X'0127')
| D*          ** 37 Bytes in segment 1
| DNUMBYTES1        2  INZ(X'0025')
| D*          ** Reserved field
| DRESR7            2  INZ(X'00')
| D*          ** Segment 1 access control pointer
| DBITMAP1A         8
| DBITMAP1B         8
| DBITMAP1C         8
| DBITMAP1D         8
| DBITMAP1E         5
| D*          ** Starting bit of segment 2 is 512 (Hex 200)
| DSTART2           2  INZ(X'0200')
| D*          ** Ending bit of segment 2 is 575 (Hex 23F)
| DEND2             2  INZ(X'023F')
| D*          ** 8 Bytes in segment 2
| DNUMBYTES2        2  INZ(X'0008')
| D*          ** Reserved field
| DRESR8            2  INZ(X'0000')
| D*          ** Segment 2 access control points
| DBITMAP2          8
| D*
| D*          *-----*
| D*          * DEFAULT expressed in ASCII *
| D*          *-----*
| DDEFAULT          S          8  INZ(X'44454641554C5420')
| D*
| D*****
| D* Prototype for Access_Control_Initialize (CSUAACI)
| D*****
| DCSUAACI          PR
| DRETCODE          9B 0
| DRSNCODE          9B 0
| DEXTDTALEN        9B 0
| DEXTDTA           4
| DRARRAYCT         9B 0
| DRARRAY           16
| DVRBDTALEN1       9B 0

```

```

|       DVRBDA1                278
|       DVRBDTALEN2           9B 0
|       DVRBDA2                335
|       D*
|       D*****
|       D* Prototype for One_Way_Hash (CSNBOWH)
|       D*****
|       DCSNBOWH              PR
|       DRETCOD                9B 0
|       DRSNCOD                9B 0
|       DEXTDTALN             9B 0
|       DEXTDT                 4
|       DRARRYCT              9B 0
|       DRARRY                 16
|       DTXTLEN               9B 0
|       DTXT                   20
|       DCHNVCTLEN            9B 0
|       DCHNVCT               128
|       DSHLEN                9B 0
|       DSH                    20
|       D*
|       D*-----
|       D*          ** Declares for sending messages to the
|       D*          ** job log using the QMHSNDPM API
|       D*-----
|       DMSG                   S          64    DIM(3) CTDATA PERRCD(1)
|       DMSGLENGTH            S          9B 0  INZ(64)
|       D                      DS
|       DMSGTEXT               1          75
|       DSAPI                  1          7
|       DFAILRETC              41        44
|       DFAILRSNC              46        49
|       DMESSAGEID             S          7    INZ(' ')
|       DMESSAGEFILE           S          21  INZ(' ')
|       DMSGKEY                 S          4    INZ(' ')
|       DMSGTYPE               S          10  INZ('*INFO ')
|       DSTACKENTRY            S          10  INZ('* ')
|       DSTACKCOUNTER          S          9B 0 INZ(2)
|       DERRCODE               DS
|       DBYTESIN               1          4B 0 INZ(0)
|       DBYTESOUT              5          8B 0 INZ(0)
|       C*
|       C*****
|       C* START OF PROGRAM *
|       C* * * * *
|       C*-----
|       C* Set up roles in verb data 2 *
|       C*-----
|       C* Set ROLE name (ROLE1)
|       C      MOVEL 'ROLE1 ' ROLE
|       C* *-----
|       C* * Set Access Control Points for ROLE1
|       C* *
|       C* * DEFAULT is authorized to all access control points
|       C* * except for the following:
|       C* * 0x0018 - Load 1st part of Master Key
|       C* * 0x0019 - Combine Master Key Parts
|       C* * 0x001A - Set Master Key
|       C* * 0x0020 - Generate Random Master Key
|       C* * 0x0032 - Clear New Master Key Register
|       C* * 0x0033 - Clear Old Master Key Register
|       C* * 0x00D6 - Translate CV
|       C* * 0x0110 - Set Clock
|       C* * 0x0111 - Reinitialize device
|       C* * 0x0112 - Initialize access control system
|       C* * 0x0113 - Change user profile expiration date
|       C* * 0x0114 - Change authentication data (eg. passphrase)

```

```

| C* * 0x0115 - Reset password failure count
| C* * 0x0116 - Read Public Access Control Information
| C* * 0x0117 - Delete user profile
| C* * 0x0118 - Delete role
| C* * 0x0119 - Load Function Control Vector
| C* * 0x011A - Clear Function Control Vector
| C* * 0x011B - Force User Logoff
| C* * 0x0200 - Register PKA Public Key Hash
| C* * 0x0201 - Register PKA Public Key, with cloning
| C* * 0x0202 - Register PKA Public Key
| C* * 0x0203 - Delete Retained Key
| C* * 0x0204 - PKA Clone Key Generate
| C* * 0x0211 - 0x21F - Clone information - obtain 1-15
| C* * 0x0221 - 0x22F - Clone information - install 1-15
| C* *
| C* * ROLE 1 is authorized to all access control points
| C* * to which the DEFAULT role is authorized plus the following:
| C* *
| C* * 0x0018 - Load 1st part of Master Key
| C* * 0x0020 - Generate Random Master Key
| C* * 0x0032 - Clear New Master Key Register
| C* * 0x0053 - Load 1st part of PKA Master Key
| C* * 0x0060 - Clear New PKA Master Key Register
| C* * 0x0119 - Load Function Control Vector
| C* * 0x0201 - Register PKA Public Key, with cloning
| C* * 0x0202 - Register PKA Public Key
| C* * 0x0203 - Delete Retained Key
| C* * 0x0204 - PKA Clone Key Generate
| C* * 0x0211 - 0x215 - Clone information - obtain 1-5
| C* * 0x0221 - 0x225 - Clone information - install 1-5
| C* *
| C* *-----
| C          EVAL      BITMAP1A = X'0003F09D80002000'
| C          EVAL      BITMAP1B = X'8000100080000000'
| C          EVAL      BITMAP1C = X'000A8000881F7110'
| C          EVAL      BITMAP1D = X'1004031180000000'
| C          EVAL      BITMAP1E = X'FF7F004F80'
| C          EVAL      BITMAP2  = X'78007C007C00E60F'
| C*   Copy role into aggregate structure
| C          MOVEVL    ROLESTRUCT    ROLE1
| C*   Set ROLE name (ROLE2)
| C          MOVEVL    'ROLE2  '    ROLE
| C* *-----
| C* * Set Access Control Points for ROLE2
| C* *
| C* * ROLE 2 is authorized to all access control points
| C* * to which the DEFAULT role is authorized plus the following:
| C* *
| C* * 0x0019 - Combine Master Key Parts
| C* * 0x001A - Set Master Key
| C* * 0x0033 - Clear Old Master Key Register
| C* * 0x0054 - Combine PKA Master Key Parts
| C* * 0x0057 - Set PKA Master Key
| C* * 0x0061 - Clear Old Master Key Register
| C* * 0x011A - Clear Function Control Vector
| C* * 0x0200 - Register PKA Public Key Hash
| C* * 0x0201 - Register PKA Public Key, with cloning
| C* * 0x0203 - Delete Retained Key
| C* * 0x0204 - PKA Clone Key Generate
| C* * 0x0216 - 0x21A - Clone information - obtain 6-10
| C* * 0x0226 - 0x22A - Clone information - install 6-10
| C* *
| C* *-----
| C          EVAL      BITMAP1A = X'0003F07D80001000'
| C          EVAL      BITMAP1B = X'8000090040000000'
| C          EVAL      BITMAP1C = X'000A8000881F7110'
| C          EVAL      BITMAP1D = X'1004031180000000'

```



```

C          EVAL      BITMAP1E = X'FF7F002F80'
C          EVAL      BITMAP2  = X'D80003E003E0E60F'
C*   Copy role into aggregate structure
C          MOVEVL    ROLESTRUCT    ROLE2
C*   Set ROLE name (ROLE3)
C          MOVEVL    'ROLE3 '      ROLE
C* *-----*
C* * Set Access Control Points for ROLE3
C* *
C* *   ROLE 3 is authorized to all access control points
C* *   to which the DEFAULT role is authorized plus the following:
C* *
C* *   0x0110 - Set Clock
C* *   0x0111 - Reinitialize device
C* *   0x0112 - Initialize access control system
C* *   0x0113 - Change user profile expiration date
C* *   0x0114 - Change authentication data (eg. passphrase)
C* *   0x0115 - Reset password failure count
C* *   0x0116 - Read Public Access Control Information
C* *   0x0117 - Delete user profile
C* *   0x0118 - Delete role
C* *   0x011B - Force User Logoff
C* *   0x0200 - Register PKA Public Key Hash
C* *   0x0201 - Register PKA Public Key, with cloning
C* *   0x0203 - Delete Retained Key
C* *   0x0204 - PKA Clone Key Generate
C* *   0x021B - 0x21F - Clone information - obtain 11-15
C* *   0x022B - 0x22F - Clone information - install 11-15
C* *
C* *-----*
C          EVAL      BITMAP1A = X'0003F01D00000000'
C          EVAL      BITMAP1B = X'80000000C0000000'
C          EVAL      BITMAP1C = X'000A8000881F7110'
C          EVAL      BITMAP1D = X'1004021180000000'
C          EVAL      BITMAP1E = X'FF7FFF9F80'
C          EVAL      BITMAP2  = X'D800001F001FE60F'
C*   Copy role into aggregate structure
C          MOVEVL    ROLESTRUCT    ROLE3
C*-----*
C* Set up roles in verb data 1 *
C*-----*
C*   Set Profile name (SECOFR1)
C          MOVEVL    'SECOFR1 '    USERID
C*   Set Role name (ROLE1)
C          MOVEVL    'ROLE1 '      ROLENAME
C*   Hash pass-phrase for profile 1
C          SETOFF                                05
C          EVAL      TEXT = 'Is it safe'
C          Z-ADD     10          TEXTLEN
C          EXSR      HASHMSG
C 05          SETON                                LR
C*   Copy profile into aggregate structure
C          MOVEVL    PROFILESTRUCT PROF1
C*   Set Profile name (SECOFR2)
C          MOVEVL    'SECOFR2 '    USERID
C*   Set Role name (ROLE2)
C          MOVEVL    'ROLE2 '      ROLENAME
C*   Hash pass-phrase for profile 2
C          EVAL      TEXT = 'I think it is safe'
C          Z-ADD     18          TEXTLEN
C          EXSR      HASHMSG
C 05          SETON                                LR
C*   Copy profile into aggregate structure
C          MOVEVL    PROFILESTRUCT PROF2
C*   Set Profile name (SECOFR3)
C          MOVEVL    'SECOFR2 '    USERID
C*   Set Role name (ROLE3)

```

```

C          MOVEL      'ROLE3  '    ROLENAME
C*   Hash pass-phrase for profile 3
C          EVAL      TEXT = 'Is what safe'
C          Z-ADD     12          TEXTLEN
C          EXSR      HASHMSG
C   05          SETON
C*   Copy profile into aggregate structure
C          MOVEL     PROFILESTRUCT PROF3
C*-----*
C* Set the keywords in the rule array *
C*-----*
C          MOVEL     'INIT-AC '    RULEARRAY
C          MOVE      'REPLACE '    RULEARRAY
C          Z-ADD     2          RULEARRAYCNT
C*****
C* Call Access_Control_Initialize SAPI
C*****
C          CALLP     CSUAACI      (RETURNCODE:
C                                REASONCODE:
C                                EXITDATALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                VERBDATALEN1:
C                                VERBDATA1:
C                                VERBDATALEN2:
C                                VERBDATA2)
C* *-----*
C* * Check the return code *
C* *-----*
C   RETURNCODE   IFGT     0
C* *-----*
C* * Send failure message *
C* *-----*
C          MOVEL     MSG(1)      MSGTEXT
C          MOVE      RETURNCODE   FAILRETC
C          MOVE      REASONCODE   FAILRSNC
C          MOVEL     'CSUAACI'   SAPI
C          EXSR      SNDMSG
C          RETURN
C          ELSE
C* *-----*
C* * Send success message *
C* *-----*
C          MOVEL     MSG(2)      MSGTEXT
C          EXSR      SNDMSG
C          ENDIF
C*-----*
C* Change the Default Role *
C*-----*
C*   Set the Role name
C          MOVEL     DEFAULT     ROLE
C* *-----*
C* * Set Access Control Points for DEFAULT
C* *
C* *-----*
C          EVAL     BITMAP1A = X'0003F01D00000000'
C          EVAL     BITMAP1B = X'8000000000000000'
C          EVAL     BITMAP1C = X'000A8000881F7110'
C          EVAL     BITMAP1D = X'1004021180000000'
C          EVAL     BITMAP1E = X'FF7F406B80'
C          EVAL     BITMAP2  = X'000000000000E60F'
C*   Copy role into aggregate structure
C          MOVEL     ROLESTRUCT   ROLE1
C*
C*   Set the new verb data 2 length

```

LR

```

C          Z-ADD      117          VERBDATALEN2
C*
C*   Set the verb data 1 length to 0 (No profiles)
C          Z-ADD      0          VERBDATALEN1
C*   Change the number of roles to 1
C          Z-ADD      1          NUMROLES
C
C*****
C* Call Access_Control_Initialize SAPI
C*****
C          CALLP      CSUAACI      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     VERBDATALEN1:
C                                     VERBDATA1:
C                                     VERBDATALEN2:
C                                     VERBDATA2)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      0
C*   *-----*
C*   * Send failure message *
C*   *-----*
C          MOVE      MSG(1)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC
C          MOVE      REASONCODE  FAILRSNC
C          MOVE      'CSUAACI'   SAPI
C          EXSR      SNDMSG
C*
C          ELSE
C*   *-----*
C*   * Send success message *
C*   *-----*
C          MOVE      MSG(3)      MSGTEXT
C          EXSR      SNDMSG
C*
C          ENDIF
C          SETON                      LR
C*
C*****
C* Subroutine to send a message
C*****
C          SNDMSG      BEGSR
C          CALL        'QMHSNDPM'
C          PARM        MESSAGEID
C          PARM        MESSAGEFILE
C          PARM        MSGTEXT
C          PARM        MSGLENGTH
C          PARM        MSGTYPE
C          PARM        STACKENTRY
C          PARM        STACKCOUNTER
C          PARM        MSGKEY
C          PARM        ERRCODE
C          ENDSR
C*
C*****
C* Subroutine to Hash pass-phrase
C*****
C          HASHMSG      BEGSR
C*   *-----*
C*   * Set the keywords in the rule array *
C*   *-----*

```

```

C          MOVEL      'SHA-1  '  RULEARRAY
C          Z-ADD      1          RULEARRAYCNT
C* *-----*
C* * Call One Way Hash SAPI *
C* *-----*
C          CALLP      CSNBOWH    (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     TEXTLEN:
C                                     TEXT:
C                                     CHAINVCTLEN:
C                                     CHAINVCT:
C                                     HASHLEN:
C                                     AUTHDATA)
C* *-----*
C* * Check the return code *
C* *-----*
C          RETURNCODE  IFGT      0
C* *-----*
C* * Send failure message *
C* *-----*
C          MOVEL      MSG(1)     MSGTEXT
C          MOVE       RETURNCODE FAILRETC
C          MOVE       REASONCODE FAILRSNC
C          MOVEL      'CSNBOWH' SAPI
C          EXSR       SNDMSG
C          SETON
C          ENDIF
C*
C          ENDSR

```

05

**

CSUAACI failed with return/reason codes 9999/9999.
SECOFR1, SECOFR2, and SECOFR3 profiles were successfully created.
The Default role was successfully changed.

関連概念

25 ページの『役割を定義する』

例: コプロセッサのデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするための ILE RPG プログラム:

コプロセッサ用のデフォルトの役割ですべてのアクセス制御ポイントを使用可能にするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

D*****
D* SETDEFAULT
D*
D* Sample program to authorize the default role to all access
D* control points in the cardX.
D*
D*
D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
D*
D* This material contains programming source code for your

```

```

| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D* IBM CCA Basic Services Reference and Guide
| D* (SC31-8609) publication.
| D*
| D* Parameters: None
| D*
| D* Example:
| D* CALL PGM(SETDEFAULT)
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(SETDEFAULT) SRCFILE(SAMPLE)
| D* CRTPGM PGM(SETID) MODULE(SETDEFAULT)
| D* BNDSRVPGM(QCCA/CSUAACI)
| D*
| D* Note: Authority to the CSUAACI service program in the
| D* QCCA library is assumed.
| D*
| D* The Common Cryptographic Architecture (CCA) verbs used are
| D* Access_Control_Initialize (CSUAACI)
| D*
| D*****
| D*-----
| D* Declare variables used by CCA SAPI calls
| D*-----
| D*          ** Return code
| DRETURNCODE S          9B 0
| D*          ** Reason code
| DREASONCODE S          9B 0
| D*          ** Exit data length
| DEXITDATALEN S         9B 0
| D*          ** Exit data
| DEXITDATA S            4
| D*          ** Rule array count
| DRULEARRAYCNT S        9B 0
| D*          ** Rule array
| DRULEARRAY S           16
| D*          ** Verb data 1 length
| DVERBDATALEN1 S        9B 0 INZ(0)
| D*          ** Verb data 1
| DVERBDATA1 S            4
| D*          ** Verb data 2 length
| DVERBDATALEN2 S        9B 0 INZ(117)
| D*-----
| D* Verbddata 2 contains the aggregate role structure which
| D* in turn contains 1 role - the default role
| D*-----
| DVERBDATA2 DS          200
| D*          ** Define 1 Role
| DNUMROLES          9B 0 INZ(1)
| D*          ** Reserved field
| DRESR1             9B 0 INZ(0)
| D*          ** Version 1 struct
| DVERS              2 INZ(X'0100')
| D*          ** Length of role
| DROLELEN           2 INZ(X'006D')
| D*          ** Description of role
| DCOMMENT           20 INZ(' ')

```

```

|      D*          ** Checksum is not used
|      DCHECKSUM          2      INZ(X'0000')
|      D*          ** Reserved field
|      DRESR2            2      INZ(X'0000')
|      D*          ** Role Name is DEFAULT expressed in ASCII
|      DROLE            8      INZ(X'44454641554C5420')
|      D*          ** Authentication strength is set to 0
|      DAUTHSTRN        2      INZ(X'0000')
|      D*          ** Lower time is 00:00
|      DLWRTIMHR        1      INZ(X'00')
|      DLWRTIMMN        1      INZ(X'00')
|      D*          ** Upper time is 23:59
|      DUPRTIMHR        1      INZ(X'17')
|      DUPRTIMMN        1      INZ(X'3B')
|      D*          ** Valid days of week
|      DVALIDDOW        1      INZ(X'FE')
|      D*          ** Reserved field
|      DRESR3            1      INZ(X'00')
|      D*          ** 2 Access control points segments are defined
|      DNUMSEG          2      INZ(X'0002')
|      D*          ** Reserved field
|      DRESR4            2      INZ(X'0000')
|      D*          ** Starting bit of segment 1 is 0.
|      DSTART1          2      INZ(X'0000')
|      D*          ** Ending bit of segment 1 is 295 (Hex 127).
|      DEND1            2      INZ(X'0127')
|      D*          ** 37 Bytes in segment 1
|      DNUMBYTES1       2      INZ(X'0025')
|      D*          ** Reserved field
|      DRESR5            2      INZ(X'00')
|      D*          ** Segment 1 access control points
|      DBITMAP1A        8      INZ(X'0003F0FD80003000')
|      DBITMAP1B        8      INZ(X'80001900C0000000')
|      DBITMAP1C        8      INZ(X'000A8000882F7110')
|      DBITMAP1D        8      INZ(X'1804033180000000')
|      DBITMAP1E        5      INZ(X'FF7FFFFF80')
|      D*          ** Starting bit of segment 2 is 512 (Hex 200).
|      DSTART2          2      INZ(X'0200')
|      D*          ** Ending bit of segment 2 is 575 (Hex 23F)
|      DEND2            2      INZ(X'023F')
|      D*          ** 8 Bytes in segment 2
|      DNUMBYTES2       2      INZ(X'0008')
|      D*          ** Reserved field
|      DRESR6            2      INZ(X'0000')
|      D*          ** Segment 2 access control points
|      DBITMAP2         8      INZ(X'F8007FFF7FFFE60F')
|      D*
|      D*****
|      D* Prototype for Access_Control_Initialize (CSUAACI)
|      D*****
|      DCSUAACI          PR
|      DRETCODE          9B 0
|      DRSNCODE          9B 0
|      DEXTDTALEN        9B 0
|      DEXTDTA           4
|      DRARRAYCT         9B 0
|      DRARRAY           16
|      DVRBDTALEN1       9B 0
|      DVRBDTA1          4
|      DVRBDTALEN2       9B 0
|      DVRBDTA2          200
|      D*
|      D*-----
|      D*          ** Declares for sending messages to the
|      D*          ** job log using the QMHSNDPM API
|      D*-----
|      DMSG              S          64      DIM(2) CTDATA PERRCD(1)

```

```

DMSGLENGT     S           9B 0 INZ(64)
D              DS
DMSGTEXT      1          64
DFAILRETC     41         44
DFAILRSNC     46         49
DMESSAGEID    S           7   INZ('   ')
DMESSAGEFILE  S          21   INZ('           ')
DMSGKEY       S           4   INZ('   ')
DMSGTYPE      S          10   INZ('*INFO ')
DSTACKENTRY   S          10   INZ('*   ')
DSTACKCOUNTER S          9B 0 INZ(2)
DERRCODE      DS
DBYTESIN      1          4B 0 INZ(0)
DBYTESOUT     5          8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM
C*
C*-----*
C* Set the keywords in the rule array
C*-----*
C          MOVE    'INIT-AC '  RULEARRAY
C          MOVE    'REPLACE '  RULEARRAY
C          Z-ADD   2           RULEARRAYCNT
C*****
C* Call Access_Control_Initialize SAPI
C*****
C          CALLP   CSUAACI      (RETURNCODE:
C                               REASONCODE:
C                               EXITDATALEN:
C                               EXITDATA:
C                               RULEARRAYCNT:
C                               RULEARRAY:
C                               VERBDATALEN1:
C                               VERBDATA1:
C                               VERBDATALEN2:
C                               VERBDATA2)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      4
C* *-----*
C* * Send failure message *
C* *-----*
C          MOVE    MSG(1)      MSGTEXT
C          MOVE    RETURNCODE  FAILRETC
C          MOVE    REASONCODE  FAILRSNC
C          EXSR    SNDMSG
C*
C          ELSE
C* *-----*
C* * Send success message *
C* *-----*
C          MOVE    MSG(2)      MSGTEXT
C          EXSR    SNDMSG
C*
C          ENDIF
C*
C          SETON
C*
C*****
C* Subroutine to send a message
C*****
C          SNDMSG    BEGSR
C          CALL      'QMHSNDPM'
C          PARM      MESSAGEID
C          PARM      MESSAGEFILE

```

```

|      C                PARM                MSGTEXT
|      C                PARM                MSGLENGTH
|      C                PARM                MSGTYPE
|      C                PARM                STACKENTRY
|      C                PARM                STACKCOUNTER
|      C                PARM                MSGKEY
|      C                PARM                ERRCODE
|      C                ENDSR

```

```

| **
| CSUAACI failed with return/reason codes 9999/9999.
| The Default role was successfully set.

```

関連概念

25 ページの『役割を定義する』

例: コプロセッサ用の既存のプロファイルを変更するための ILE C プログラム:

コプロセッサ用に既存のプロファイルを変更するには、必要に応じて以下のプログラムを変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| /*-----*/
| /* Change certain fields in a user profile on the          */
| /* card. This program changes the expiration date using a new */
| /* date in the form YYYYMMDD.                               */
| /*                                                         */
| /*                                                         */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 1999           */
| /*                                                         */
| /* This material contains programming source code for your  */
| /* consideration. These examples have not been thoroughly  */
| /* tested under all conditions. IBM, therefore, cannot     */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are     */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF     */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files.                                */
| /*                                                         */
| /*                                                         */
| /* Note: Input format is more fully described in Chapter 2 of */
| /*       IBM CCA Basic Services Reference and Guide          */
| /*       (SC31-8609) publication.                            */
| /*                                                         */
| /* Parameters:                                             */
| /* none.                                                  */
| /*                                                         */
| /* Example:                                               */
| /* CALL PGM(CHG_PROF)                                     */
| /*                                                         */
| /* Note: This program assumes the card with the profile is  */
| /* already identified either by defaulting to the CRP01     */
| /* device or by being explicitly named using the           */
| /* Cryptographic_Resource_Allocate verb. Also this        */
| /* device must be varied on and you must be authorized    */
| /* to use this device description.                         */
| /*                                                         */
| /*                                                         */

```



```

| /* The Common Cryptographic Architecture (CCA) verb used is      */
| /* Access_Control_Initialization (CSUAACI).                      */
| /*                                                                */
| /* Use these commands to compile this program on the system:    */
| /* ADDLIB LIB(QCCA)                                             */
| /* CRTCPGM MODULE(CHG_PROF) SRCFILE(SAMPLE)                     */
| /* CRTCPGM PGM(CHG_PROF) MODULE(CHG_PROF)                       */
| /*      BNDSRVPGM(QCCA/CSUAACI)                                 */
| /*                                                                */
| /* Note: Authority to the CSUAACI service program in the       */
| /*      QCCA library is assumed.                                */
| /*                                                                */
| /* The Common Cryptographic Architecture (CCA) verb used is      */
| /* Access_Control_Initialization (CSUAACI).                      */
| /*                                                                */
| /*-----*/
|
| #include "csucincl.h"      /* header file for CCA Cryptographic */
|                          /* Service Provider                */
|
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
| #include <decimal.h>
|
| /*-----*/
| /* standard return codes                                         */
| /*-----*/
|
| #define ERROR    -1
| #define OK       0
| #define WARNING  4
|
| int main(int argc, char *argv[])
| {
|     /*-----*/
|     /* standard CCA parameters                                   */
|     /*-----*/
|
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[8];
|     long rule_array_count = 1;
|
|     /*-----*/
|     /* fields unique to this sample program                     */
|     /*-----*/
|
|     long verb_data_length;
|     char * verb_data;
|     long verb_data_length2;
|     char * verb_data2;
|
|     memcpy(rule_array,"CHGEXPDT",8);      /* set rule array keywords */
|
|     verb_data_length = 8;
|
|     verb_data = "SECOFR1 ";              /* set the profile name    */
|
|     verb_data_length2 = 8;
|
|     verb_data2 = "20010621";            /* set the new date       */

```

```

| /* invoke verb to change the expiration date in specified profile */
|
| CSUAACI( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (char *)rule_array,
|         &verb_data_length,
|         verb_data,
|         &verb_data_length2,
|         verb_data2);
|
| if ( (return_code == OK) | (return_code == WARNING) )
| {
|     printf("Profile expiration date was changed successfully");
|     printf(" with return/reason codes ");
|     printf("%ld/%ld\n\n", return_code, reason_code);
|     return(OK);
| }
|
| else
| {
|     printf("Change of expiration date failed with return/");
|     printf("reason codes ");
|     printf(" %ld/%ld\n\n", return_code, reason_code);
|     return(ERROR);
| }
| }
|

```

関連概念

25 ページの『プロファイルを定義する』

例: コプロセッサ用の既存のプロファイルを変更するための ILE RPG プログラム:

コプロセッサ用に既存のプロファイルを変更するには、必要に応じて以下のプログラムを変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| D*****
| D* CHG_PROF
| D*
| D* Change certain fields in a user profile on the
| D* card. This program changes the expiration date using a new
| D* date in the form YYYYMMDD.
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.

```

```

| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D* IBM CCA Basic Services Reference and Guide
| D* (SC31-8609) publication.
| D*
| D* Parameters: Profile
| D*
| D* Example:
| D* CALL PGM(CHG_PROF) PARM(PROFILE)
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(CHG_PROF) SRCFILE(SAMPLE)
| D* CRTPGM PGM(CHG_PROF) MODULE(CHG_PROF)
| D* BNDDIR(QCCA/QC6BNDDIR)
| D*
| D* Note: Authority to the CSUAACI service program in the
| D* QCCA library is assumed.
| D*
| D* The Common Cryptographic Architecture (CCA) verbs used are
| D* Access_Control_Initialize (CSUAACI)
| D*
| D* This program assumes the card with the profile is
| D* already identified either by defaulting to the CRP01
| D* device or by being explicitly named using the
| D* Cryptographic_Resource_Allocate verb. Also this
| D* device must be varied on and you must be authorized
| D* to use this device description.
| D*****
| D*-----
| D* Declare variables for CCA SAPI calls
| D*-----
| D* ** Return code
| DRETURNCODE S 9B 0
| D* ** Reason code
| DREASONCODE S 9B 0
| D* ** Exit data length
| DEXITDATALEN S 9B 0
| D* ** Exit data
| DEXITDATA S 4
| D* ** Rule array count
| DRULEARRAYCNT S 9B 0
| D* ** Rule array
| DRULEARRAY S 16
| D* ** Verb data 1 length
| DVERBDATALEN1 S 9B 0 INZ(8)
| D* ** Verb data 1
| DVERBDATA1 S 8
| D* ** Verb data 2 length
| DVERBDATALEN2 S 9B 0 INZ(8)
| D* ** Verb data 2
| DVERBDATA2 S 8
| D*
| D*
| D*****
| D* Prototype for Access_Control_Initialize (CSUAACI)
| D*****
| DCSUAACI PR
| DRETCODE 9B 0
| DRSNCODE 9B 0
| DEXTDTALEN 9B 0
| DEXTDTA 4
| DRARRAYCT 9B 0
| DRARRAY 16
| DVERBDTALEN1 9B 0
| DVERBDTA1 8
| DVERBDTALEN2 9B 0

```

```

DVRBDA2                8
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
MSG          S          75    DIM(2) CTDATA PERRCD(1)
MSGLENGTH   S          9B 0  INZ(75)
D           DS
MSGTEXT      1          75
DFAILRETC   41         44
DFAILRSNC   46         49
DMESSAGEID  S          7     INZ('      ')
DMESSAGEFILE S         21    INZ('          ')
MSGKEY       S          4     INZ('      ')
MSGTYPE      S         10    INZ('*INFO  ')
DSTACKENTRY S         10    INZ('*      ')
DSTACKCOUNTER S        9B 0  INZ(2)
DERRCODE     DS
DBYTESIN     1          4B 0  INZ(0)
DBYTESOUT    5          8B 0  INZ(0)
C*****
C* START OF PROGRAM *
C* *
C*-----*
C* Parameter is profile to be changed. *
C*-----*
C   *ENTRY      PLIST
C               PARM                VERBDA1
C*-----*
C* Set the keywords in the rule array *
C*-----*
C               MOVEL   'CHGEXPDT'  RULEARRAY
C               Z-ADD   1          RULEARRAYCNT
C*-----*
C* Set new expiration date *
C*-----*
C               MOVEL   '20061231'  VERBDA2
C*-----*
C* Call Access_Control_Initialize SAPI *
C*-----*
C               CALLP   CSUAACI      (RETURNCODE:
C                                   REASONCODE:
C                                   EXITDATALEN:
C                                   EXITDATA:
C                                   RULEARRAYCNT:
C                                   RULEARRAY:
C                                   VERBDALEN1:
C                                   VERBDA1:
C                                   VERBDALEN2:
C                                   VERBDA2)
C*-----*
C* Check the return code *
C*-----*
C   RETURNCODE  IFGT      0
C* *-----*
C* * Send error message *
C* *-----*
C               MOVE    MSG(1)      MSGTEXT
C               MOVE    RETURNCODE  FAILRETC
C               MOVE    REASONCODE  FAILRSNC
C               EXSR    SNDMSG
C*
C               ELSE
C* *-----*
C* * Send success message *
C* *-----*

```

```

|      C          MOVE      MSG(2)      MSGTEXT
|      C          EXSR      SNDMSG
|      C*
|      C          ENDIF
|      C*
|      C          SETON                      LR
|      C*
|      C*****
|      C* Subroutine to send a message
|      C*****
|      C      SNDMSG      BEGSR
|      C          CALL          'QMHSNDPM'
|      C          PARM          MESSAGEID
|      C          PARM          MESSAGEFILE
|      C          PARM          MSGTEXT
|      C          PARM          MSGLENGTH
|      C          PARM          MSGTYPE
|      C          PARM          STACKENTRY
|      C          PARM          STACKCOUNTER
|      C          PARM          MSGKEY
|      C          PARM          ERRCODE
|      C          ENDSR
|      C*
|
|      **
|      CSUAACI failed with return/reason codes 9999/9999'
|      The request completed successfully

```

関連概念

25 ページの『プロファイルを定義する』

環境 ID およびクロックの設定

暗号化コプロセッサは、鍵トークンを作成したコプロセッサを検査します。4758 コプロセッサは、時刻と日付のスタンプを行うため、およびプロファイルがログオンできるかどうかを制御するためにクロックを使用します。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

環境 ID (EID): コプロセッサは、EID を識別子として保管します。最も簡単、かつ迅速に EID を設定するには、暗号化コプロセッサ構成のための Web ベースのユーティリティーを使用します。このユーティリティーには、<http://server-name:2001> の「System Tasks」ページからアクセスできます。このユーティリティーには、コプロセッサが初期化されていない状態の場合に使用される、基本構成ウィザードが含まれています。コプロセッサが初期設定済みである場合は、「構成の管理」、次に「属性 (Attributes)」をクリックし、EID を設定します。

独自のアプリケーションを作成して、EID を設定することもできます。それには、Cryptographic_Facility_Control (CSUACFC) API verb を使用します。参考のために、2 つのプログラム例が提供されています。そのうちの 1 つは ILE C で作成されており、もう 1 つは ILE RPG で作成されています。どちらのプログラムも実行する機能は同じです。

暗号化コプロセッサは、コプロセッサが作成するすべての PKA 鍵トークンに EID をコピーします。EID を使用すると、コプロセッサは、別のコプロセッサが作成した鍵ではなく、そのコプロセッサが作成した鍵を識別することができます。

関連資料

64 ページの『例: コプロセッサに環境 ID を設定するための ILE C プログラム』

コプロセッサで環境 ID を設定するには、必要に応じて以下のプログラムを変更してください。

| 66 ページの『例: コプロセッサに環境 ID を設定するための ILE RPG プログラム』
| コプロセッサで環境 ID を設定するには、必要に応じて以下のプログラムを変更してください。

| **クロック:** コプロセッサは、クロック・カレンダーを使用して時刻と日付を記録し、プロファイルがロ
| グオンできるかどうかを判断します。デフォルトの時刻は、グリニッジ標準時刻 (GMT) です。その機能に
| より、設定するためのデフォルトの役割の機能を削除する前に、コプロセッサ内部でクロックを設定する
| 必要があります。

| 最も簡単、かつ迅速にクロックを設定するには、暗号化コプロセッサ構成のための Web ベースのユーテ
| ィリティーを使用します。このユーティリティーには、<http://server-name:2001> の「System Tasks」ページか
| らアクセスできます。このユーティリティーには、コプロセッサが初期化されていない状態の場合に使用
| される、基本構成ウィザードが含まれています。暗号化コプロセッサが初期設定済みである場合は、「構
| 成の管理」、次に「属性 (Attributes)」をクリックし、クロックを設定します。

| 独自のアプリケーションを作成して、クロックを設定することもできます。これを行うには、
| Cryptographic_Facility_Control (CSUACFC) API verb を使用します。

| 関連資料

| 69 ページの『例: コプロセッサにクロックを設定するための ILE C プログラム』
| コプロセッサでクロックを設定するには、必要に応じて以下のプログラムを変更してください。

| 71 ページの『例: コプロセッサにクロックを設定するための ILE RPG プログラム』
| コプロセッサでクロックを設定するには、必要に応じて以下のプログラムを変更してください。

| 例: コプロセッサに環境 ID を設定するための ILE C プログラム:

| コプロセッサで環境 ID を設定するには、必要に応じて以下のプログラムを変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

```
| /*-----*/  
| /* Set the environment ID on the card, based on a */  
| /* 16-byte sample value defined in this program. */  
| /* */  
| /* */  
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999 */  
| /* */  
| /* This material contains programming source code for your */  
| /* consideration. These examples have not been thoroughly */  
| /* tested under all conditions. IBM, therefore, cannot */  
| /* guarantee or imply reliability, serviceability, or function */  
| /* of these program. All programs contained herein are */  
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */  
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */  
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */  
| /* these programs and files. */  
| /* */  
| /* */  
| /* Note: Input format is more fully described in Chapter 2 of */  
| /* IBM CCA Basic Services Reference and Guide */  
| /* (SC31-8609) publication. */  
| /* */  
| /* Parameters: */  
| /* none. */  
| /* */  
| /* Example: */  
| /* CALL PGM(SETVID) */  
| /* */  
| /* */
```

```

| /* Note: This program assumes the device to use is          */
| /* already identified either by defaulting to the CRP01    */
| /* device or by being explicitly named using the          */
| /* Cryptographic_Resource_Allocate verb. Also this       */
| /* device must be varied on and you must be authorized   */
| /* to use this device description.                       */
| /*                                                       */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA)                                       */
| /* CRTCMOD MODULE(SETVID) SRCFILE(SAMPLE)                 */
| /* CRTPGM PGM(SETVID) MODULE(SETVID)                     */
| /* BNSRVPGM(QCCA/CSUACFC)                                */
| /*                                                       */
| /* Note: Authority to the CSUACFC service program in the  */
| /* QCCA library is assumed.                               */
| /*                                                       */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Cryptographic_Facilites_Control (CSUACFC).            */
| /*                                                       */
| /*-----*/
|
| #include "csucincl.h" /* header file for CCA Cryptographic */
| /* Service Provider */
|
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
| #define WARNING 4
|
| int main(int argc, char *argv[])
|
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[2][8];
|     long rule_array_count = 2;
|
|     /*-----*/
|     /* fields unique to this sample program */
|     /*-----*/
|
|     long verb_data_length;
|     char * verb_data = "SOME ID data 16@";
|
|     /* set keywords in the rule array */
|
|     memcpy(rule_array, "ADAPTER1SET-EID ", 16);
|
|     verb_data_length = 16;
|
|     /* invoke the verb to set the environment ID */

```

```

|     CSUACFC(&return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             &rule_array_count,
|             (char *)rule_array,
|             &verb_data_length,
|             verb_data);
|
|     if ( (return_code == OK) | (return_code == WARNING) )
|     {
| printf("Environment ID was successfully set with ");
|
| printf("return/reason codes %ld/%ld\n\n", return_code, reason_code);
|
| return(OK);
|     }
|
|     else
|     {
| printf("An error occurred while setting the environment ID.\n");
|
| printf("Return/reason codes %ld/%ld\n\n", return_code, reason_code);
|
| return(ERROR);
|     }
| }

```

関連概念

63 ページの『環境 ID (EID)』

例: コプロセッサに環境 ID を設定するための ILE RPG プログラム:

コプロセッサで環境 ID を設定するには、必要に応じて以下のプログラムを変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* SETEID
| D*
| D* Set the environment ID on the card, based on a
| D* 16-byte sample value defined in this program.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D*       IBM CCA Basic Services Reference and Guide
| D*       (SC31-8609) publication.
| D*
| D* Parameters: None
| D*
| D* Example:
| D*   CALL PGM(SETOID)

```



```

D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(SETUID) SRCFILE(SAMPLE)
D* CRTPGM PGM(SETUID) MODULE(SETUID)
D* BNDSRVPGM(QCCA/CSUACFC)
D*
D* Note: Authority to the CSUACFC service program in the
D* QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Cryptographic_Facilty_Control (CSUACFC)
D*
D*****
D*-----
D* Declare variables for CCA SAPI calls
D*-----
D*          ** Return code
DRETURNCODE S          9B 0
D*          ** Reason code
DREASONCODE S          9B 0
D*          ** Exit data length
DEXITDATALEN S         9B 0
D*          ** Exit data
DEXITDATA S           4
D*          ** Rule array count
DRULEARRAYCNT S        9B 0
D*          ** Rule array
DRULEARRAY S           16
D*          ** Verb data length
DVERBDATALEN S         9B 0
D*          ** Verb data
DVERBDATA S           16 INZ('Card ID 01234567')
D*
D*
D*****
D* Prototype for Cryptographic_Facilty_Control (CSUACFC)
D*****
DCSUACFC PR
DRETCODE 9B 0
DRSNCODE 9B 0
DEXTDTALEN 9B 0
DEXTDTA 4
DRARRAYCT 9B 0
DRARRAY 16
DVRBDTALEN 9B 0
DVRBDTA 16
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG S 75 DIM(2) CTDATA PERRCD(1)
DMSGLENGHTH S 9B 0 INZ(75)
D DS
DMSGTEXT 1 80
DFAILRETC 41 44
DFAILRSNC 46 49
DMESSAGEID S 7 INZ(' ')
DMESSAGEFILE S 21 INZ(' ')
DMSGKEY S 4 INZ(' ')
DMSGTYPE S 10 INZ('*INFO ')
DSTACKENTRY S 10 INZ('* ')
DSTACKCOUNTER S 9B 0 INZ(2)
DERRCODE DS
DBYTESIN 1 4B 0 INZ(0)
DBYTESOUT 5 8B 0 INZ(0)
C*

```

```

C*****
C* START OF PROGRAM *
C* *
C*-----*
C* Set the keyword in the rule array *
C*-----*
C          MOVEL      'ADAPTER1'    RULEARRAY
C          MOVE       'SET-EID '    RULEARRAY
C          Z-ADD      2              RULEARRAYCNT
C*-----*
C* Set the verb data length to 16 *
C*-----*
C          Z-ADD      16              VERBDATALEN
C*****
C* Call Cryptographic Facility Control SAPI *
C*****
C          CALLP      CSUACFC        (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     VERBDATALEN:
C                                     VERBDATA)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT          4
C*          *-----*
C*          * Send error message *
C*          *-----*
C          MOVEL      MSG(1)         MSGTEXT
C          MOVE       RETURNCODE     FAILRETC
C          MOVE       REASONCODE     FAILRSNC
C          EXSR      SNDMSG
C*
C          ELSE
C*          *-----*
C*          * Send success message *
C*          *-----*
C          MOVE       MSG(2)         MSGTEXT
C          EXSR      SNDMSG
C*
C          ENDIF
C*
C          SETON
C*
C*****
C* Subroutine to send a message
C*****
C          SNDMSG     BEGSR
C          CALL       'QMHSNDPM'
C          PARM      MESSAGEID
C          PARM      MESSAGEFILE
C          PARM      MSGTEXT
C          PARM      MSGLENGTH
C          PARM      MSGTYPE
C          PARM      STACKENTRY
C          PARM      STACKCOUNTER
C          PARM      MSGKEY
C          PARM      ERRCODE
C          ENDSR

```

**

CSUACFC failed with return/reason codes 9999/9999.
The Environment ID was successfully set.

関連概念

例: コプロセッサにクロックを設定するための ILE C プログラム:

コプロセッサでクロックを設定するには、必要に応じて以下のプログラムを変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

/*-----*/
/* Set the clock on the card, based on a string from */
/* the command line. The command line string must be of */
/* form YYYYMMDDHHMSSWW, where WW is the day of week (01 */
/* means Sunday and 07 means Saturday). */
/* */
/* */
/* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999 */
/* */
/* This material contains programming source code for your */
/* consideration. These examples have not been thoroughly */
/* tested under all conditions. IBM, therefore, cannot */
/* guarantee or imply reliability, serviceability, or function */
/* of these program. All programs contained herein are */
/* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
/* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
/* these programs and files. */
/* */
/* */
/* Note: Input format is more fully described in Chapter 2 of */
/* IBM CCA Basic Services Reference and Guide */
/* (SC31-8609) publication. */
/* */
/* Parameters: */
/* char * new time 16 characters */
/* */
/* Example: */
/* CALL PGM(SETCLOCK) PARM('1999021011375204') */
/* */
/* Note: This program assumes the device to use is */
/* already identified either by defaulting to the CRP01 */
/* device or by being explicitly named using the */
/* Cryptographic_Resource_Allocate verb. Also this */
/* device must be varied on and you must be authorized */
/* to use this device description. */
/* */
/* Use these commands to compile this program on the system: */
/* ADDLIB LIB(QCCA) */
/* CRTCMOD MODULE(SETCLOCK) SRCFILE(SAMPLE) */
/* CRTPGM PGM(SETCLOCK) MODULE(SETCLOCK) */
/* BNDSRVPGM(QCCA/CSUACFC) */
/* */
/* Note: Authority to the CSUACFC service program in the */
/* QCCA library is assumed. */
/* */
/* The Common Cryptographic Architecture (CCA) verb used is */
/* Cryptographic_Facilities_Control (CSUACFC). */
/* */
/*-----*/

#include "csucincl.h" /* header file for CCA Cryptographic */
/* Service Provider */
#include <stdio.h>
#include <string.h>

```

```

| #include <stdlib.h>
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR    -1
| #define OK       0
| #define WARNING  4
|
| void help(void)
| {
|     printf("\n\nThis program loads the time and date into the card.\n");
|
|     printf("It requires a single command line parameter containing the %n");
|
|     printf("new date and time in the form YYYYMMDDHHMMSSWW, where WW is the%n");
|     printf("day of the week, 01 meaning Sunday and 07 meaning Saturday.\n\n");
|
| }
|
| int main(int argc, char *argv[])
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[2][8];
|     long rule_array_count = 2;
|
|     /*-----*/
|     /* fields unique to this sample program */
|     /*-----*/
|
|     long verb_data_length;
|     char * verb_data;
|
|     if (argc != 2)
|     {
|         help();
|
|         return(ERROR);
|     }
|
|     if (strlen(argv[1]) != 16)
|     {
|         printf("Your input string is not the right length.");
|
|         help();
|
|         return(ERROR);
|     }
|
|     /* set keywords in the rule array */
|
|     memcpy(rule_array,"ADAPTER1SETCLOCK",16);

```

```

|     verb_data_length = 16;
|
|     /* copy keyboard input for new time                               */
|
|     verb_data = argv[1];
|
|     /* Set the clock to the time the user gave us                       */
|
|     CSUACFC( &return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             &rule_array_count,
|             (char *)rule_array,
|             &verb_data_length,
|             verb_data);
|
|     if ( (return_code == OK) | (return_code == WARNING) )
|     {
|         printf("Clock was successfully set.\nReturn/");
|
|         printf("reason codes %ld/%ld\n\n", return_code, reason_code);
|
|         return(OK);
|     }
|
|     else
|     {
|         printf("An error occurred while setting the clock.\nReturn");
|
|         printf("/reason codes %ld/%ld\n\n", return_code, reason_code);
|
|         return(ERROR);
|     }
| }

```

関連概念

64 ページの『クロック』

例: コプロセッサにクロックを設定するための ILE RPG プログラム:

コプロセッサでクロックを設定するには、必要に応じて以下のプログラムを変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* SETCLOCK
| D*
| D* Set the clock on the card, based on a string from
| D* the command line. The command line string must be of
| D* form YYYYMMDDHHMSSWW, where WW is the day of week (01
| D* means Sunday and 07 means Saturday).
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.

```

```

D*
D*
D* Note: Input format is more fully described in Chapter 2 of
D*   IBM CCA Basic Services Reference and Guide
D*   (SC31-8609) publication.
D*
D* Parameters:
D*   char * new time 16 characters
D*
D* Example:
D*   CALL PGM(SETCLOCK) PARM('2000061011375204')
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(SETCLOCK) SRCFILE(SAMPLE)
D* CRTPGM PGM(SETCLOCK) MODULE(SETCLOCK)
D*   BNDSRVPGM(QCCA/CSUACFC)
D*
D* Note: Authority to the CSUACFC service program in the
D*   QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Cryptographic_Facilty_Control (CSUACFC)
D*
D*****
D*-----
D* Declare variables for CCA SAPI calls
D*-----
D*          ** Return code
DRETURNCODE S          9B 0
D*          ** Reason code
DREASONCODE S          9B 0
D*          ** Exit data length
DEXITDATALEN S         9B 0
D*          ** Exit data
DEXITDATA S           4
D*          ** Rule array count
DRULEARRAYCNT S        9B 0
D*          ** Rule array
DRULEARRAY S           16
D*          ** Verb data length
DVERBDATALEN S         9B 0
D*          ** Verb data
DVERBDATA S           16
D*
D*****
D* Prototype for Cryptographic_Facilty_Control (CSUACFQ)
D*****
DCSUACFC PR
DRETCODE          9B 0
DRSNCODE          9B 0
DEXTDTALEN        9B 0
DEXTDTA           4
DRARRAYCT         9B 0
DRARRAY           16
DVRBDTALEN        9B 0
DVRBDTA           16
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG S           75 DIM(6) CTDATA PERRCD(1)
DMSGLENGTH S      9B 0 INZ(75)
D DS
DMSGTEXT          1 80
DFAILRETC         41 44
DFAILRSNC         46 49

```

```

| DMESSAGEID      S          7  INZ('      ')
| DMESSAGEFILE    S          21 INZ('          ')
| DMSGKEY         S          4  INZ('      ')
| DMSGTYPE        S          10 INZ('*INFO  ')
| DSTACKENTRY     S          10 INZ('*      ')
| DSTACKCOUNTER   S          9B 0 INZ(2)
| DERRCODE        DS
| DBYTESIN        1          4B 0 INZ(0)
| DBYTESOUT       5          8B 0 INZ(0)
| C*
| C*****
| C* START OF PROGRAM *
| C* *
| C   *ENTRY      PLIST
| C               PARM                VERBDATA
| C* *
| C*-----*
| C* Check the number of parameters passed in *
| C*-----*
| C               IF      (%PARMS < 1)
| C* *-----*
| C* * Send message describing the format of the parameter *
| C* *-----*
| C               MOVEL    MSG(3)      MSGTEXT
| C               EXSR    SNDMSG
| C               MOVEL    MSG(4)      MSGTEXT
| C               EXSR    SNDMSG
| C               MOVEL    MSG(5)      MSGTEXT
| C               EXSR    SNDMSG
| C               MOVEL    MSG(6)      MSGTEXT
| C               EXSR    SNDMSG
| C               RETURN
| C               ENDIF
| C*
| C*-----*
| C* Set the keyword in the rule array *
| C*-----*
| C               MOVEL    'ADAPTER1'  RULEARRAY
| C               MOVE    'SETCLOCK'   RULEARRAY
| C               Z-ADD    2            RULEARRAYCNT
| C*-----*
| C* Set the verb data length to 16 *
| C*-----*
| C               Z-ADD    16          VERBDATALEN
| C*****
| C* Call Cryptographic Facility Control SAPI */
| C*****
| C               CALLP    CSUACFC      (RETURNCODE:
| C                                     REASONCODE:
| C                                     EXITDATALEN:
| C                                     EXITDATA:
| C                                     RULEARRAYCNT:
| C                                     RULEARRAY:
| C                                     VERBDATALEN:
| C                                     VERBDATA)
| C*-----*
| C* Check the return code *
| C*-----*
| C               RETURNCODE  IFGT      4
| C* *-----*
| C* * Send error message *
| C* *-----*
| C               MOVEL    MSG(1)      MSGTEXT
| C               MOVE    RETURNCODE   FAILRETC
| C               MOVE    REASONCODE   FAILRSNC
| C               EXSR    SNDMSG
| C*

```

```

|      C          ELSE
|      C*          *-----*
|      C*          * Send success message *
|      C*          *-----*
|      C              MOVE      MSG(2)      MSGTEXT
|      C              EXSR      SNDMSG
|      C*
|      C              ENDIF
|      C*
|      C              SETON                      LR
|      C*
|      C*****
|      C* Subroutine to send a message
|      C*****
|      C      SNDMSG      BEGSR
|      C              CALL      'QMHSNDPM'
|      C              PARM              MESSAGEID
|      C              PARM              MESSAGEFILE
|      C              PARM              MSGTEXT
|      C              PARM              MSGLENGTH
|      C              PARM              MSGTYPE
|      C              PARM              STACKENTRY
|      C              PARM              STACKCOUNTER
|      C              PARM              MSGKEY
|      C              PARM              ERRCODE
|      C              ENDSR
|
|      **
|      CSUACFC failed with return/reason codes 9999/9999.
|      The request completed successfully.
|      This program loads the time and date into the card.
|      It requires a single command line parameter containing the
|      new date and time in the form YYYYMMDDHHMSSWW, where WW is the
|      day of the week, 01 meaning Sunday and 07 meaning Saturday.

```

関連概念

64 ページの『クロック』

機能制御ベクトルのロード

機能制御ベクトルは、鍵を作成するために使用する鍵長を暗号化コプロセッサに通知します。機能制御ベクトルをロードしないと、すべての暗号機能を実行することができません。

役割およびプロファイルを作成および定義したあとに、暗号化コプロセッサ用の機能制御ベクトル (FCV) をロードする必要があります。機能制御ベクトルがないと、コプロセッサはすべての暗号操作を実行することができません。

機能制御ベクトルとは、IBM が提供する、ファイルに保管された、デジタル署名済みの値です。i5/OS Option 35 をインストールすると、ファイルは /QIBM/ProdData/CAP/FCV.CRT というパスで、ルート・ファイル・システムに保管されます。この値を使用すると、コプロセッサ内部の暗号アプリケーションは、適用可能なインポート規則およびエクスポート規則に一致する一定レベルの暗号サービスを実行することができます。

最も簡単、かつ迅速に FCV をロードするには、暗号化コプロセッサ構成のための Web ベースのユーティリティーを使用します。このユーティリティーには、<http://server-name:2001> の「Tasks」ページからアクセスできます。このユーティリティーには、コプロセッサが初期化されていない状態の場合に使用される、基本構成ウィザードが含まれています。コプロセッサが初期設定済みである場合は、「構成の管理」、次に「属性 (Attributes)」をクリックし、FCV をロードします。

独自のアプリケーションを作成して、FCV をロードすることもできます。それには、Cryptographic_Facility_Control (CSUACFC) API verb を使用します。

| 機能制御ベクトルをクリアする方法を示すサンプル・プログラムが、他にも 2 つ提供されています。その
| うちの 1 つは ILE C で作成されており、もう 1 つは ILE RPG で作成されています。

| コプロセッサ用の機能制御ベクトルをロードすると、マスター鍵を使用して、鍵を暗号化するために使用
| するマスター鍵をロードして設定することができます。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 関連概念

| 24 ページの『役割およびプロファイルの作成と定義』

| 暗号化コプロセッサは、役割ベースのアクセス制御を使用します。役割ベースのシステムでは、コプ
| ロセッサ・ユーザーのクラスに対応する一連の役割を定義します。使用可能な役割の 1 つにユーザ
| ーをマップするために、関連したユーザー・プロファイルを定義して各ユーザーを登録できます。

| 86 ページの『マスター鍵のロードおよび設定』

| 機能制御ベクトルをロードした後で、マスター鍵をロードして設定します。マスター鍵を使用すると他
| の鍵を暗号化することができます。

| 例: 暗号化コプロセッサ用の機能制御ベクトルをロードするための ILE C プログラム:

| 暗号化コプロセッサ用の機能制御ベクトルをロードするには、必要に応じて以下のプログラムを変更して
| ください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

```
| /*-----*/  
| /* Load the Function Control Vector into the card. */  
| /* The Function Control Vector enables the cryptographic */  
| /* functions of the card and is shipped with the */  
| /* Cryptographic Access Provider products. */  
| /* */  
| /* COPYRIGHT 5769-SS1 (c) IBM Corp 1999 */  
| /* */  
| /* This material contains programming source code for your */  
| /* consideration. These examples have not been thoroughly */  
| /* tested under all conditions. IBM, therefore, cannot */  
| /* guarantee or imply reliability, serviceability, or function*/  
| /* of these programs. All programs contained herein are */  
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */  
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */  
| /* EXPRESSLY DISCLAIMED. IBM provides no program services for*/  
| /* these programs and files. */  
| /* */  
| /* Note: The Function Control Vector is stored in an IFS */  
| /* file owned by the system. The format of this */  
| /* vector is described in an appendix of the */  
| /* IBM CCA Basic Services Reference and Guide */  
| /* (SC31-8609) publication. */  
| /* */  
| /* Parameters: */  
| /* none. */  
| /* */  
| /* Example: */  
| /* CALL PGM(LOAD_FCV) */  
| /* */  
| /* Note: This program assumes the device you want to load is */  
| /* already identified either by defaulting to the CRP01 */  
| /* device or has been explicitly named using the */  
| /* Cryptographic_Resource_Allocate verb. Also this */
```

```

| /*      device must be varied on and you must be authorized */
| /*      to use this device description.                      */
| /*                                                         */
| /* Use the following commands to compile this program:      */
| /* ADDLIB LIB(QCCA)                                         */
| /* CRTCMOD MODULE(LOAD_FCV) SRCFILE(SAMPLE) SYSIFCOPT(*IFSIO) */
| /* CRTPGM  PGM(LOAD_FCV) MODULE(LOAD_FCV) +                */
| /*          BNDSRVPGM(QCCA/CSUACFC)                         */
| /*                                                         */
| /* Note: Authority to the CSUACFC service program in the   */
| /*       QCCA library is assumed.                          */
| /*                                                         */
| /* Common Cryptographic Architecture (CCA) verbs used:    */
| /* Cryptographic_Facility_Control (CSUACFC)                */
| /*                                                         */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include <decimal.h>
| #include "csucincl.h"          /* header file for CCA Cryptographic
|                               Service Provider          */
|
| /*-----*/
| /* function to translate ASCII to EBCDIC and/or EBCDIC to ASCII */
| /*-----*/
|
| #pragma linkage(QDCXLATE, OS, nowiden)
| void QDCXLATE(decimal(5,0)*,
|              char *,
|              char *,
|              char *);
|
| int main(void)
| {
|
| /*-----*/
| /* standard return codes                                     */
| /*-----*/
|
| #define ERROR -1
| #define OK    0
|
| /*-----*/
| /* standard CCA parameters                                 */
| /*-----*/
|
| long return_code;
| long reason_code;
| long exit_data_length;
| char exit_data[2];
| char rule_array[4][8];
| long rule_array_count;
|
| /*-----*/
| /* fields unique to this sample program                    */
| /*-----*/
|
| long verb_data_length;
| char *verb_data;
| char buffer[1000];
| char description[81];
| decimal(5,0) descr_length = 80;
| int num_bytes;
| FILE *fcv;

```

```

| /*-----*/
| /* retrieve FCV from IBM supplied file */
| /*-----*/
| fcv = fopen("/QIBM/ProdData/CAP/FCV.CRT", "rb");
| if (fcv==NULL)
| {
|     printf("Function Control Vector file not available\n\n");
|     return ERROR;          /* File not found or not authorized */
| }
|
| num_bytes = fread(buffer,1,1000,fcv);
| fclose(fcv);
|
| if (num_bytes != 802)
| {
|     printf("Function Control Vector file has wrong size\n\n");
|     return ERROR;          /* Incorrect number of bytes read */
| }
|
| /*-----*/
| /* extract fields in FCV needed by card */
| /* Note: use offsets and lengths from CCA publication listed earlier */
| /*-----*/
|
| memcpy(description, &buffer[390],80);
| description[80] = 0;
| QDCXLATE(&descr_length, description, "QEBCDIC ", "QSYS ");
| printf("Loading Function Control Vector: %s\n",description);
|
| verb_data_length = 204;
| verb_data = &buffer[470];
|
| rule_array_count = 2;
| memcpy((char*)rule_array,"ADAPTER1LOAD-FCV",16);
|
| /*-----*/
| /* Load the card with the FCV just retrieved */
| /*-----*/
| CSUACFC(&return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (char*)rule_array,
|         &verb_data_length,
|         verb_data);
|
| if (return_code != 0)
| {
|     printf("Function Control Vector rejected for reason %d/%d\n\n",
|           return_code, reason_code);
|     return ERROR;          /* Operation failed. */
| }
| else
| {
|     printf("Loading Function Control Vector succeeded\n\n");
|     printf("SAPI returned %ld/%ld\n\n", return_code, reason_code);
|     return OK;
| }
| }

```

例: コプロセッサ用の機能制御ベクトルをロードするための ILE RPG プログラム:

コプロセッサ用の機能制御ベクトルをロードするには、必要に応じて以下のプログラムを変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

```
| D*****  
| D* LOAD_FCV  
| D*  
| D* Load the Function Control Vector into the card.  
| D* The Function Control Vector enables the cryptographic  
| D* functions of the card and is shipped with the  
| D* Cryptographic Access Provider products.  
| D*  
| D* The Function Control Vector is contained within a stream  
| D* file. Before compiling and running this program, you  
| D* must copy the contents of the stream file to a database  
| D* member. An example of how to do this is shown in the  
| D* instructions below for compiling and running this program.  
| D*  
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000  
| D*  
| D* This material contains programming source code for your  
| D* consideration. These example has not been thoroughly  
| D* tested under all conditions. IBM, therefore, cannot  
| D* guarantee or imply reliability, serviceability, or function  
| D* of these programs. All programs contained herein are  
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF  
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for  
| D* these programs and files.  
| D*  
| D*  
| D* Note: Input format is more fully described in Chapter 2 of  
| D* IBM CCA Basic Services Reference and Guide  
| D* (SC31-8609) publication.  
| D*  
| D* Parameters: None  
| D*  
| D* Example:  
| D* CALL PGM(LOAD_FCV)  
| D*  
| D* Use these commands to compile this program on the system:  
| D*  
| D* CRTRPGMOD MODULE(LOAD_FCV) SRCFILE(SAMPLE)  
| D*  
| D* CRTPGM PGM(LOAD_FCV) MODULE(LOAD_FCV)  
| D* BNDSRVPGM(QCCA/CSUACFC)  
| D*  
| D* Note: Authority to the CSUACFC service program in the  
| D* QCCA library is assumed.  
| D*  
| D* The Common Cryptographic Architecture (CCA) verbs used are  
| D* Cryptographic_Facilty_Control (CSUACFC)  
| D*  
| D*****  
| D*-----  
| D* Declare variables used by CCA SAPI calls  
| D*-----  
| D* ** Return code  
| DRETURNCODE S 9B 0  
| D* ** Reason code  
| DREASONCODE S 9B 0  
| D* ** Exit data length  
| DEXITDATALEN S 9B 0  
| D* ** Exit data  
| DEXITDATA S 4  
| D* ** Rule array count  
| DRULEARRAYCNT S 9B 0  
| D* ** Rule array
```

```

| DRULEARRAY      S          16
| D*              ** Verb data length
| DVERBDATALEN   S          9B 0 INZ(204)
| D*              ** Verb data
| DVERBDATA      S          204
| D*-----
| D* Declare variables for working with files
| D*-----
| D*              ** File descriptor
| DFILED         S          9B 0
| D*              ** File path
| DPATH          S          80   INZ('/QIBM/ProdData/CAP/FCV.CRT')
| D*              ** Open Flag - Open for Read only
| DOFLAGR       S          10I 0 INZ(1)
| D*              ** Structure of Funciton control vector file
| DFLD1         DS
| DFLDDTA        S          802
| DDESCR         S          391 470
| DFNCCTLVCT     S          471 674
| D*              ** Length of data read from file
| DINLEN         S          9B 0
| D*              ** Declares for calling QDCXLATE API
| DXLTTBL       S          10   INZ('QEBCDIC ')
| DTBLLIB       S          10   INZ('QSYS  ')
| DDESCLEN      S          5P 0 INZ(80)
| D*              ** Index into a string
| DINDEX        S          5B 0
| D*              ** Variable to hold temporary character value
| DCHAR         S          1
| D*
| D*****
| D* Prototype for Cryptographic_Facilty_Control (CSUACFC)
| D*****
| DCSUACFC       PR
| DRETCODE       S          9B 0
| DRSNCODE       S          9B 0
| DEXTDTALEN     S          9B 0
| DEXTDTA        S          4
| DRARRAYCT     S          9B 0
| DRARRAY        S          16
| DVRBDTALEN    S          9B 0
| DVRBDTA       S          204
| D*
| D*****
| D* Prototype for open()
| D*****
| D* value returned = file descriptor (OK), -1 (error)
| Dopen          PR          9B 0 EXTPROC('open')
| D* path name of file to be opened.
| D              128   OPTIONS(*VARSIZE)
| D* Open flags
| D              9B 0 VALUE
| D* (OPTIONAL) mode - access rights
| D              10U 0 VALUE OPTIONS(*NOPASS)
| D* (OPTIONAL) codepage
| D              10U 0 VALUE OPTIONS(*NOPASS)
| D*
| D*****
| D* Prototype for read()
| D*****
| D* value returned = number of bytes actually read, or -1
| Dread         PR          9B 0 EXTPROC('read')
| D* File descriptor returned from open()
| D              9B 0 VALUE
| D* Input buffer
| D              2500  OPTIONS(*VARSIZE)
| D* Length of data to be read

```

```

D                                9B 0 VALUE
D*
D*****
D* Prototype for close()
D*****
D*   value returned = 0 (OK), or -1
Dclose          PR              9B 0 EXTPROC('close')
D*   File descriptor returned from open()
D                                9B 0 VALUE
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG           S              80   DIM(4) CTDATA PERRCD(1)
DMSGLENGTH    S              9B 0 INZ(80)
D              DS
DMSGTEXT      1              80
DFAILRETC    41              44
DFAILRSNC    46              49
DMESSAGEID   S              7   INZ('      ')
DMESSAGEFILE S              21  INZ('          ')
DMSGKEY      S              4   INZ('      ')
DMSGTYPE     S              10  INZ('*INFO ')
DSTACKENTRY  S              10  INZ('*      ')
DSTACKCOUNTER S             9B 0 INZ(2)
DERRCODE     DS
DBYTESIN     1              4B 0 INZ(0)
DBYTESOUT    5              8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM *
C* *
C*-----*
C* Open the FCV file *
C*-----*
C* *-----*
C* ** Null terminate path name *
C* *-----*
C          EVAL          %SUBST(PATH:27:1) = X'00'
C* *-----*
C* * Open the file *
C* *-----*
C          EVAL          FILED = open(PATH: OFLAGR)
C* *-----*
C* * Check if open worked *
C* *-----*
C  FILED          IFEQ          -1
C* *-----*
C* * Open failed, send an error message *
C* *-----*
C          MOVEL          MSG(1)          MSGTEXT
C          EXSR          SNDMSG
C          RETURN
C*
C          ENDIF
C* *-----*
C* * Open worked, read the FCV, and close the file *
C* *-----*
C          Z-ADD          802          INLEN
C          EVAL          INLEN = read(FILED: FLDDTA: INLEN)
C          CALLP          close          (FILED)
C*
C* *-----*
C* * Check if read operation was OK *
C* *-----*
C  INLEN          IFEQ          -1

```

```

C          MOVEL      MSG(2)      MSGTEXT
C          EXSR       SNDMSG
C          RETURN
C          ENDIF
C*-----*
C* Copy the FCV to the verb data parameter.      *
C*-----*
C          MOVEL      FNCCTLVCT   VERBDATA
C*-----*
C* Convert description to EBCDIC and display it   *
C*-----*
C          CALL      'QDCXLATE'
C          PARM      DESCLEN
C          PARM      DESCR
C          PARM      XLTTBL
C          PARM      TBLLIB
C          MOVEL     DESCR      MSGTEXT
C          Z-ADD     80         INDEX
C*-----*
C*   Replace trailing null characters in description *
C*   with space characters.                          *
C*-----*
C          SETOFF
C          DOU        *IN50
C          EVAL      CHAR = %SUBST(MSGTEXT:INDEX:1)
C          CHAR      IFNE      X'00'
C          SETON
C          ELSE
C          EVAL      %SUBST(MSGTEXT:INDEX:1) = ' '
C          INDEX     SUB       1          INDEX
C          INDEX     IFEQ      0
C          SETON
C          ENDIF
C          ENDIF
C          ENDDO
C          EXSR      SNDMSG
C*-----*
C* Set the keywords in the rule array              *
C*-----*
C          MOVEL     'ADAPTER1'   RULEARRAY
C          MOVE      'LOAD-FCV'   RULEARRAY
C          Z-ADD     2            RULEARRAYCNT
C*****
C* Call Cryptographic Facility Control SAPI      */
C*****
C          CALLP     CSUACFC      (RETURNCODE:
C                                REASONCODE:
C                                EXITDATALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                VERBDATALEN:
C                                VERBDATA)
C* *-----*
C* * Check the return code *
C* *-----*
C          RETURNCODE IFGT      0
C* *-----*
C* * Send failure message *
C* *-----*
C          MOVEL     MSG(3)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC
C          MOVE      REASONCODE  FAILRSNC
C          EXSR      SNDMSG
C*
C          ELSE

```

```

| C*
| C* *-----*
| C* * Send success message *
| C* *-----*
| C          MOVEL      MSG(4)      MSGTEXT
| C          EXSR       SNDMSG
| C          ENDIF
| C*
| C          SETON                               LR
| C*
| C*****
| C* Subroutine to send a message
| C*****
| C      SNDMSG      BEGSR
| C          CALL      'QMHSNDPM'
| C          PARM      MESSAGEID
| C          PARM      MESSAGEFILE
| C          PARM      MSGTEXT
| C          PARM      MSGLENGTH
| C          PARM      MSGTYPE
| C          PARM      STACKENTRY
| C          PARM      STACKCOUNTER
| C          PARM      MSGKEY
| C          PARM      ERRCODE
| C          ENDSR
| **
| Error trying to open FCV file.
| Error reading data from FCV file.
| CSUACFC failed with return/reason codes 9999/9999.
| The Function Control Vector was successfully loaded.

```

例: コプロセッサから機能制御ベクトルをクリアするための ILE C プログラム:

コプロセッサ用の機能制御ベクトルをクリアするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| /*-----*/
| /* Clear the Function Control Vector from the card. */
| /* The Function Control Vector enables the cryptographic */
| /* functions of the card. Clearing it from the */
| /* disabled the cryptographic functions. */
| /* */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 2000 */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or */
| /* functions of these program. All programs contained */
| /* herein are provided to you "AS IS". THE IMPLIED */
| /* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A */
| /* PARTICULAR PURPOSE ARE ARE EXPRESSLY DISCLAIMED. IBM */
| /* provides no program services for these programs and files.*/
| /* */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* none. */
| /* */
| /* Example: */

```



```

| /* CALL PGM(CLEARFCV) */
| /* */
| /* */
| /* Use the following command to compile this program: */
| /* CRTCMOD MODULE(CLEARFCV) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(CLEARFCV) MODULE(CLEARFCV) */
| /* BNDSRVPGM(QCCA/CSUACFC) */
| /* */
| /* Common Cryptographic Architecture (CCA) verbs used: */
| /* - Cryptographic_Facility_Control (CSUACFC) */
| /* */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h"
|
| void main(void)
| {
|     long return_code;
|     long reason_code;
|     long exit_data_length;
|     char exit_data[2];
|     char rule_array[4][8];
|     long rule_array_count;
|     long verb_data_length;
|     char *verb_data;
|     char buffer[4];
|
|     /*-----*/
|     /* No verb data is needed for this option. */
|     /*-----*/
|     verb_data_length = 0;
|     verb_data = buffer;
|
|     /*-----*/
|     /* Rule array has two elements or rule array keywords */
|     /*-----*/
|     rule_array_count = 2;
|     memcpy((char*)rule_array,"ADAPTER1CLR-FCV ",16);
|
|     /*-----*/
|     /* Clear the Function control vector from the card */
|     /*-----*/
|     CSUACFC(&return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             &rule_array_count,
|             (char*)rule_array,
|             &verb_data_length,
|             verb_data);
|
|     if (return_code != 0)
|         printf("Operation failed: return code %d : reason code %d %n",
|               return_code, reason_code);
|     else
|         printf("FCV is successfullly cleared%n");
| }

```

例: コプロセッサから機能制御ベクトルをクリアするための ILE RPG プログラム:

コプロセッサ用の機能制御ベクトルをクリアするには、必要に応じて以下のプログラム例を変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

```
| D*****  
| D* CLEARFCV  
| D*  
| D* Clear the Function Control Vector from the card.  
| D* The Function Control Vector enables the cryptographic  
| D* functions of the card. Clearing it from the  
| D* disabled the cryptographic functions.  
| D*  
| D*  
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000  
| D*  
| D* This material contains programming source code for your  
| D* consideration. These example has not been thoroughly  
| D* tested under all conditions. IBM, therefore, cannot  
| D* guarantee or imply reliability, serviceability, or function  
| D* of these programs. All programs contained herein are  
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF  
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for  
| D* these programs and files.  
| D*  
| D*  
| D* Note: Input format is more fully described in Chapter 2 of  
| D* IBM CCA Basic Services Reference and Guide  
| D* (SC31-8609) publication.  
| D*  
| D* Parameters: None  
| D*  
| D* Example:  
| D* CALL PGM(CLEARFCV)  
| D*  
| D* Use these commands to compile this program on the system:  
| D* CRTRPGMOD MODULE(CLEARFCV) SRCFILE(SAMPLE)  
| D* CRTPGM PGM(CLEARFCV) MODULE(CLEARFCV)  
| D* BNDSRVPGM(QCCA/CSUACFC)  
| D*  
| D* Note: Authority to the CSUACFC service program in the  
| D* QCCA library is assumed.  
| D*  
| D* The Common Cryptographic Architecture (CCA) verbs used are  
| D* Cryptographic_Facilty_Control (CSUACFC)  
| D*  
| D*****  
| D*-----  
| D* Declare variables used on CCA SAPI calls  
| D*-----  
| D* ** Return code  
| DRETURNCODE S 9B 0  
| D* ** Reason code  
| DREASONCODE S 9B 0  
| D* ** Exit data length  
| DEXITDATALEN S 9B 0  
| D* ** Exit data  
| DEXITDATA S 4  
| D* ** Rule array count  
| DRULEARRAYCNT S 9B 0  
| D* ** Rule array  
| DRULEARRAY S 16  
| D* ** Verb data length  
| DVERBDATALEN S 9B 0  
| D* ** Verb data  
| DVERBDATA S 16  
| D*  
| D*
```

```

D*****
D* Prototype for Cryptographic_Facilty_Control (CSUACFQ)
D*****
DCSUACFC          PR
DRETCODE          9B 0
DRSNCODE          9B 0
DEXTDTALEN       9B 0
DEXTDTA           4
DRARRAYCT        9B 0
DRARRAY           16
DVRBDTALEN       9B 0
DVRBDTA          10
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG             S          75  DIM(2) CTDATA PERRCD(1)
DMSGLENGTH      S          9B 0  INZ(75)
D               DS
DMSGTEXT         1          75
DFAILRETC        41         44
DFAILRSNC        46         49
D*          ** Variables required for the QMHSNDPM API
DMESSAGEID       S          7  INZ(' ')
DMESSAGEFILE     S          21  INZ(' ')
DMSGKEY          S          4  INZ(' ')
DMSGTYPE         S          10  INZ('*INFO ')
DSTACKENTRY     S          10  INZ('* ')
DSTACKCOUNTER   S          9B 0  INZ(2)
DERRCODE         DS
DBYTESIN         1          4B 0  INZ(0)
DBYTESOUT        5          8B 0  INZ(0)
D*
C*****
C* START OF PROGRAM *
C* *
C*-----*
C* Set the keyword in the rule array *
C*-----*
C          MOVE      'ADAPTER1'  RULEARRAY
C          MOVE      'CLR-FCV '  RULEARRAY
C          Z-ADD     2           RULEARRAYCNT
C*-----*
C* Set the verb data length to 0 *
C*-----*
C          Z-ADD     0           VERBDATALEN
C*-----*
C* Call Cryptographic Facilty Control SAPI
C*-----*
C          CALLP    CSUACFC      (RETURNCODE:
C                                REASONCODE:
C                                EXITDTALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                VERBDATALEN:
C                                VERBDATA)
C*-----*
C* Check the return code
C*-----*
C          RETURNCODE  IFGT      0
C*          *-----*
C*          * Send a failure message *
C*          *-----*
C          MOVE      MSG(1)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC

```

```

|      C          MOVE      REASONCODE  FAILRSNC
|      C          EXSR      SNDMSG
|      C*
|      C          ELSE
|      C*          *-----*
|      C*          * Send a Success message *
|      C*          *-----*
|      C          MOVE      MSG(2)      MSGTEXT
|      C          EXSR      SNDMSG
|      C*
|      C          ENDIF
|      C*
|      C          SETON                      LR
|      C*
|      C*****
|      C* Subroutine to send a message
|      C*****
|      C      SNDMSG      BEGSR
|      C          CALL          'QMHSNDPM'
|      C          PARM          MESSAGEID
|      C          PARM          MESSAGEFILE
|      C          PARM          MSGTEXT
|      C          PARM          MSGLENGTH
|      C          PARM          MSGTYPE
|      C          PARM          STACKENTRY
|      C          PARM          STACKCOUNTER
|      C          PARM          MSGKEY
|      C          PARM          ERRCODE
|      C          ENDSR
|      C*
|      **
|      CSUACFC failed with return/reason codes 9999/9999'
|      The request completed successfully

```

マスター鍵のロードおよび設定

機能制御ベクトルをロードした後で、マスター鍵をロードして設定します。マスター鍵を使用すると他の鍵を暗号化することができます。

機能制御ベクトルをロードした後で、マスター鍵をロードして設定します。コプロセッサは、マスター鍵を使用してすべての操作鍵を暗号化します。マスター鍵は、コプロセッサのセキュア・モジュール内にあるクリア (暗号化されていない) に保管されている特殊な鍵暗号化鍵です。コプロセッサは、マスター鍵を使用して他の鍵を暗号化し、コプロセッサ外部にこれらの鍵を保管できるようにします。マスター鍵は、少なくとも 2 つの 168 ビット・パーツを排他論理和演算して得られる 168 ビットの鍵です。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

関連概念

74 ページの『機能制御ベクトルのロード』

機能制御ベクトルは、鍵を作成するために使用する鍵長を暗号化コプロセッサに通知します。機能制御ベクトルをロードしないと、すべての暗号機能を実行することができません。

マスター鍵のロード: マスター鍵には、新規、現行[®]、および旧という 3 つのレジスタがあります。新規マスター鍵レジスタは、作成中のマスター鍵を一時的に保持するために使用されます。このマスター鍵は、鍵の暗号化には使用されません。現行マスター鍵レジスタは、新規に生成、インポート、あるいは再暗号化された鍵を暗号化するために現在使用中のマスター鍵を保持します。旧マスター鍵レジスタは、以前のマスター鍵を保持します。このレジスタに保持されているマスター鍵は、マスター鍵が変更された後に鍵を回復するために使用されます。マスター鍵をロードすると、コプロセッサはそれを、新規マスター鍵レジスタに配置します。マスター鍵を設定するまで、そこに残っています。

1 | マスター鍵を作成およびロードするには、セキュリティーの必要性に基づいて次の 3 つの方法の中から 1
1 | つを使用します。

1 | • 最初の鍵パーツとその後の鍵パーツを別個にロードして全体として鍵の情報を分割しておく。この方法
1 | は、安全性が最も低い方法ですが、各鍵パーツを別個の個体に与えることによってセキュリティーを向
1 | 上させることができます。

1 | • ランダムに鍵を生成することにより、鍵に関する人間の知識を排除する。この方法は、マスター鍵をロ
1 | ードするには最も安全性の高い方法ですが、コピーを作成するためランダムに生成されたマスター鍵を
1 | 2 番目の暗号化コプロセッサに複製する必要があります。

1 | • 別のコプロセッサから複製して、既に存在するマスター鍵を使用する。

1 | 関連情報

1 | IBM PCI 暗号化コプロセッサ・ドキュメンテーション・ライブラリー (IBM PCI Cryptographic
1 | Coprocessor documentation library)

1 | **マスター鍵の設定:** マスター鍵を設定すると、現行マスター鍵レジスター内にある鍵が旧マスター鍵レジ
1 | スターに移動します。次に、新規マスター鍵レジスター内にあるマスター鍵が、現行マスター鍵レジスター
1 | に移動します。

1 | **注:** マスター鍵によって暗号化されたデータを取り出すには、常にそのマスター鍵のバックアップ・コピー
1 | を取っておくことが不可欠です。例えば、1 枚の紙に書いておいて、そのバックアップ・コピーを適切
1 | なセキュリティー上の注意を払って保管するようにしてください。または、別のコプロセッサにマス
1 | ター鍵の複製を作成してください。

1 | 最も簡単、かつ迅速にマスター鍵をロードおよび設定するには、暗号化コプロセッサ構成のための Web
1 | ベースのユーティリティーを使用します。このユーティリティーには、<http://server-name:2001> の「System
1 | Tasks」ページからアクセスできます。このユーティリティーには、コプロセッサが初期化されていない
1 | 状態の場合に使用される、基本構成ウィザードが含まれています。暗号化コプロセッサが初期設定済みで
1 | ある場合は、「構成の管理」、次に「マスター鍵 (Master keys)」をクリックし、マスター鍵をロードし、
1 | 設定します。

1 | 独自のアプリケーションを作成して、マスター鍵をロードして設定することもできます。それには、
1 | Master_Key_Process (CSNBMKP) API verb を使用します。

1 | 関連資料

1 | 88 ページの『例: 暗号化コプロセッサにマスター鍵をロードするための ILE C プログラム』
1 | 暗号化コプロセッサに新規マスター鍵をロードするには、必要に応じて以下のプログラム例を変更し
1 | てください。

1 | 90 ページの『例: 暗号化コプロセッサにマスター鍵をロードするための ILE RPG プログラム』
1 | 暗号化コプロセッサに新規マスター鍵をロードするには、必要に応じて以下のプログラム例を変更し
1 | てください。

1 | **鍵の再暗号化:** マスター鍵を設定する際、アクセスできなくなることを防ぐために、以前のマスター鍵で
1 | 暗号化されたすべての鍵を再暗号化する必要があります。再暗号化は、マスター鍵を変更および設定する前
1 | に行います。

1 | 暗号化コプロセッサ構成のための Web ベースのユーティリティーを使用して、鍵ストア内の鍵を再暗号
1 | 化することができます。このユーティリティーには、<http://server-name:2001> の「System Tasks」ページから
1 | アクセスできます。暗号化コプロセッサは初期設定済みでなければなりません。「構成の管理」、さらに
1 | 「DES 鍵」とクリックして DES 鍵を再暗号化するか、「PKA 鍵」をクリックして PKA 鍵を再暗号化し
1 | ます。

| 鍵ストアにない鍵がある場合、あるいは鍵を再暗号化するための独自のアプリケーションを作成する場合
| は、 Key_Token_Change (CSNBKTC) または PKA_Key_Token_Change (CSNDKTC) API verb を使用しま
| す。

| 参考のために、プログラム例が 1 つ提供されています。

| 関連資料

| 94 ページの『例: 暗号化コプロセッサの鍵を再暗号化するための ILE C プログラム』
| 暗号化コプロセッサ用の鍵を再暗号化するには、必要に応じて以下のプログラムを変更してくださ
| い。

| 例: 暗号化コプロセッサにマスター鍵をロードするための ILE C プログラム:

| 暗号化コプロセッサに新規マスター鍵をロードするには、必要に応じて以下のプログラム例を変更して
| ください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
| ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これ
| らのプログラム例を修正して使用することをお勧めします。

```
| /*-----*/  
| /* Load a new master key on the card. */  
| /* */  
| /* */  
| /* COPYRIGHT 5769-SS1, 5722-SS1 (C) IBM CORP. 1999, 2000 */  
| /* */  
| /* This material contains programming source code for your */  
| /* consideration. These examples have not been thoroughly */  
| /* tested under all conditions. IBM, therefore, cannot */  
| /* guarantee or imply reliability, serviceability, or function */  
| /* of these program. All programs contained herein are */  
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */  
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */  
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */  
| /* these programs and files. */  
| /* */  
| /* */  
| /* */  
| /* Parameters: */  
| /* OPTION (FIRST, MIDDLE, LAST, CLEAR, SET) */  
| /* KEYPART (24 bytes entered in hex -> X'01F7C4....') */  
| /* Required for FIRST, MIDDLE, and LAST */  
| /* */  
| /* Example: */  
| /* CALL PGM(LOAD_KM) */  
| /* (FIRST X'0123456789ABCDEFEDCBA98765432100123456789ABCDEF') */  
| /* */  
| /* Note: This program assumes the device to use is */  
| /* already identified either by defaulting to the CRP01 */  
| /* device or by being explicitly named using the */  
| /* Cryptographic_Resource_Allocate verb. Also this */  
| /* device must be varied on and you must be authorized */  
| /* to use this device description. */  
| /* */  
| /* */  
| /* Use these commands to compile this program on the system: */  
| /* ADDLIB LIB(QCCA) */  
| /* CRTCMOD MODULE(LOAD_KM) SRCFILE(SAMPLE) */  
| /* CRTPGM PGM(LOAD_KM) MODULE(LOAD_KM) */
```

```

| /*      BNDSRVPGM(QCCA/CSNBKMP QCCA/CSNBRNG)          */
| /*      */
| /* Note: Authority to the CSNBKMP and CSNBRNG service programs */
| /*      in the QCCA library is assumed.                */
| /*      */
| /* The main Common Cryptographic Architecture (CCA) verb used */
| /* is Master_Key_Process (CSNBKMP).                    */
| /*      */
| /*-----*/
|
| #include "csucincl.h"      /* header file for CCA Cryptographic */
|                             /* Service Provider                    */
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR    -1
| #define OK       0
| #define WARNING  4
|
| int main(int argc, char *argv[])
|
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[2][8];
|     long rule_array_count = 1;
|
|     /*-----*/
|     /* parameters unique to this program */
|     /*-----*/
|     char keypart[24];          /* Dummy parm for SET and CLEAR */
|
|     /*-----*/
|     /* Process the parameters */
|     /*-----*/
|     if (argc < 2)
|     {
|         printf("Option parameter must be specified.\n");
|         return(ERROR);
|     }
|
|     if (argc < 3 && memcmp(argv[1],"CLEAR",5) != 0 &&
|         memcmp(argv[1],"SET",3) != 0)
|     {
|         printf("KeyPart parameter must be specified.\n");
|         return(ERROR);
|     }
|
|     /*-----*/
|     /* Set the keywords in the rule array */
|     /*-----*/
|     memset(rule_array,' ',8);
|     memcpy(rule_array,argv[1],
|            (strlen(argv[1]) > 8) ? 8 : strlen(argv[1]));

```

```

| /*-----*/
| /* Call Master Key Process SAPI */
| /*-----*/
| CSNBMP( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char *)rule_array,
|         (argc == 3) ? argv[2] : keypart);
|
| /*-----*/
| /* Check the return code and display the results */
| /*-----*/
| if ( (return_code == OK) | (return_code == WARNING) )
| {
|     printf("Request was successful with return/reason codes: %d/%d %n",
|           return_code, reason_code);
|     return(OK);
| }
| else
| {
|     printf("Request failed with return/reason codes: %d/%d %n",
|           return_code, reason_code);
|     return(ERROR);
| }
| }

```

関連概念

87 ページの『マスター鍵の設定』

例: 暗号化コプロセッサにマスター鍵をロードするための ILE RPG プログラム:

暗号化コプロセッサに新規マスター鍵をロードするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| D*****
| D* LOAD_KM
| D*
| D* Load a new master key on the card.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of

```



```

D*      IBM CCA Basic Services Reference and Guide
D*      (SC31-8609) publication.
D*
D* Parameters:
D*  OPTION      (FIRST, MIDDLE, LAST, CLEAR, SET)
D*  KEYPART     (24 bytes entered in hex -> X'01F7C4...')
D*              Required for FIRST, MIDDLE, and LAST
D*
D* The master key is loaded in 3 or more parts. Specify FIRST
D* when loading the first part, MIDDLE when loading all parts
D* between the first and the last, and LAST when loading the final
D* part of the master key.
D*
D* As the master key parts are entered, they are Exclusively OR'ed
D* with the current contents of the master key register. After the
D* last master key, if the contents do not have odd parity in every
D* byte, a non-zero return/reason code will be returned. In order
D* to ensure that the final result has odd parity, each key part
D* should have odd parity in every byte. This is assuming that there
D* is an odd number of key parts. (If there is an even number of
D* key parts, then one of the key parts should have even parity).
D*
D* A byte has odd parity if it contains:
D*   an odd parity nibble : 1, 2, 4, 7, 8, B, D, or E   AND
D*   an even parity nibble: 0, 3, 5, 6, 9, A, C, or F.
D*
D* For example 32, A4, 1F, and 75 are odd parity bytes because
D*              they contain both an odd parity and an even parity
D*              nibble.
D*
D*              05, 12, 6C, and E7 are even parity bytes because
D*              they contain either two even parity nibbles or
D*              two odd parity nibbles.
D*
D* The New master key register must be empty before the first part
D* of a master key can be entered. Use CLEAR to ensure that the
D* New master key register is empty before loading the master key
D* parts.
D*
D* After loading the master key, use SET to move the master key from
D* the New-master-key register to the Current-master-key register.
D* Cryptographic keys are encrypted under the master key in the
D* the Current-master-key register.
D*
D* Example:
D*  CALL PGM(LOAD_KM) (CLEAR)
D*
D*  CALL PGM(LOAD_KM)
D*    (FIRST X'0123456789ABCDEFEDCBA98765432100123456789ABCDEF')
D*
D*  CALL PGM(LOAD_KM)
D*    (MIDDLE X'1032A873458010F7EF3438373132F1F2F4F8B3CDCDCDCEF1')
D*
D*  CALL PGM(LOAD_KM)
D*    (LAST X'2040806789ABCDEFEDC3434346432100123456789FEDCBA')
D*
D*  CALL PGM(LOAD_KM) (SET)
D*
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(LOAD_KM) SRCFILE(SAMPLE)
D* CRTPGM  PGM(LOAD_KM) MODULE(LOAD_KM)
D*        BNDSRVPGM(QCCA/CSNBMP)
D*
D* Note: Authority to the CSNBMP service program in the
D*       QCCA library is assumed.

```

```

D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Master_Key_Process (CSNBMP)
D*
D*****
D*-----
D* Declare variables for CCA SAPI calls
D*-----
D*          ** Return code
DRETURNCODE  S          9B 0
D*          ** Reason code
DREASONCODE  S          9B 0
D*          ** Exit data length
DEXITDATALEN S          9B 0
D*          ** Exit data
DEXITDATA    S          4
D*          ** Rule array count
DRULEARRAYCNT S        9B 0
D*          ** Rule array
DRULEARRAY   S          16
D*          ** Option (Rule Array Keyword)
DOPTION      S          8
D*          ** Master key part parameter on program
DMASTERKEYPART S        24
D*          ** Master key part parameter on CSNBMP
DKEYPART     S          24  INZ(*ALLX'00')
D*
D*****
D* Prototype for Master_Key_Process (CSNBMP)
D*****
DCSNBMP      PR
DRETCODE     9B 0
DRSNCODE     9B 0
DEXTDTALEN   9B 0
DEXTDTA      4
DRARRAYCT    9B 0
DRARRAY      16
DMSTRKEY     24  OPTIONS(*NOPASS)
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG         S          75  DIM(2) CTDATA PERRCD(1)
DMSGLENGTH   S          9B 0 INZ(75)
D            DS
DMSGTEXT     1          75
DFAILRETC    41         44
DFAILRSNC    46         49
DMESSAGEID   S          7   INZ(' ')
DMESSAGEFILE S          21  INZ(' ')
DMSGKEY      S          4   INZ(' ')
DMSGTYPE     S          10  INZ(*INFO ')
DSTACKENTRY  S          10  INZ(* ')
DSTACKCOUNTER S        9B 0 INZ(2)
DERRCODE     DS
DBYTESIN     1          4B 0 INZ(0)
DBYTESOUT    5          8B 0 INZ(0)
D*
C*****
C* START OF PROGRAM *
C* * * * *
C  *ENTRY      PLIST
C              PARM          OPTION
C              PARM          MASTERKEYPART
C* * * * *
C*-----*

```

```

C* Set the keyword in the rule array *
C*-----*
C          MOVEL   OPTION   RULEARRAY
C          Z-ADD   1        RULEARRAYCNT
C*
C*-----*
C* Check for FIRST, MIDDLE, or LAST *
C*-----*
C  OPTION   IFEQ   'FIRST'
C  OPTION   OREQ   'MIDDLE'
C  OPTION   OREQ   'LAST'
C* *-----*
C* * Copy keypart parameter *
C* *-----*
C          MOVEL   MASTERKEYPART  KEYPART
C          ENDIF
C*
C*-----*
C* Call Master Key Process SAPI *
C*-----*
C          CALLP   CSNBMP   (RETURNCODE:
C                          REASONCODE:
C                          EXITDATALEN:
C                          EXITDATA:
C                          RULEARRAYCNT:
C                          RULEARRAY:
C                          KEYPART)
C*-----*
C* Check the return code *
C*-----*
C  RETURNCODE  IFGT   0
C* *-----*
C* * Send error message *
C* *-----*
C          MOVE   MSG(1)   MSGTEXT
C          MOVE   RETURNCODE  FAILRETC
C          MOVE   REASONCODE  FAILRSNC
C          EXSR   SNDMSG
C*
C          ELSE
C* *-----*
C* * Send success message *
C* *-----*
C          MOVE   MSG(2)   MSGTEXT
C          EXSR   SNDMSG
C*
C          ENDIF
C*
C          SETON
C*
C*****
C* Subroutine to send a message
C*****
C  SNDMSG      BEGSR
C              CALL      'QMHSNDPM'
C              PARM      MESSAGEID
C              PARM      MESSAGEFILE
C              PARM      MSGTEXT
C              PARM      MSGLENGTH
C              PARM      MSGTYPE
C              PARM      STACKENTRY
C              PARM      STACKCOUNTER
C              PARM      MSGKEY
C              PARM      ERRCODE
C              ENDSR

```

LR

```
| C*
| **
| CSNBKMP failed with return/reason codes 9999/9999
| The request completed successfully
```

関連概念

87 ページの『マスター鍵の設定』

例: 暗号化コプロセッサの鍵を再暗号化するための ILE C プログラム:

暗号化コプロセッサ用の鍵を再暗号化するには、必要に応じて以下のプログラムを変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
| /*-----*/
| /* Description: Re-enciphers key store files using the current */
| /*              master key.                               */
| /*                                                    */
| /* COPYRIGHT      5769-SS1 (c) IBM Corp 1999          */
| /*                                                    */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot    */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these programs. All programs contained herein are  */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF    */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files.                             */
| /*                                                    */
| /* Parameters:                                           */
| /* char * keysto_type, choices are "DES" or "PKA"        */
| /*              (If omitted, the default is "PKA".)     */
| /* Examples:                                             */
| /* CALL PGM(REN_KEYSTO) PARM(DES)                       */
| /* CALL PGM(REN_KEYSTO)                                 */
| /*                                                    */
| /* Note: The CCA verbs used in the this program are more fully */
| /* described in the IBM CCA Basic Services Reference */
| /* and Guide (SC31-8609) publication.                   */
| /*                                                    */
| /* Note: This program assumes the card you want to use is */
| /* already identified either by defaulting to the CRP01 */
| /* device or has been explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description.                     */
| /*                                                    */
| /* This program also assumes the key store file you will */
| /* use is already identified either by being specified on */
| /* the cryptographic device or has been explicitly named */
| /* using the Key_Store_Designate verb. Also you must be */
| /* authorized to update records in this file.          */
| /*                                                    */
| /* Use the following commands to compile this program:  */
| /* ADDLIB LIB(QCCA)                                     */
| /* CRTMOD MODULE(REN_KEYSTO) SRCFILE(SAMPLE)          */
| /* CRTPGM PGM(REN_KEYSTO) MODULE(REN_KEYSTO)         */
| /* BNDSRVPGM(QCCA/CSNBKTC QCCA/CSNBKRL)              */
```

```

| /*          QCCA/CSNDKTC QCCA/CSNDKRL          */
| /*          */
| /* Note: authority to the CSNDKTC, CSNDKRL, CSNBKTC, and CSNBKRL */
| /* service programs in the QCCA library is assumed.          */
| /*          */
| /* Common Cryptographic Architecture (CCA) verbs used:      */
| /*   PKA_Key_Token_Change (CSNDKTC)                          */
| /*   DES_Key_Token_Change (CSNBKTC)                          */
| /*   PKA_Key_Record_List (CSNDKRL)                           */
| /*   DES_Key_Record_List (CSNBKRL)                           */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h"          /* header file for CCA Cryptographic
|                               Service Provider          */
|
| /* Define the acceptable file types */
| #define PKA 1
| #define DES 0
|
| int re_encipher(FILE *key_rec, long rec_length, int key_type);
|
| int main(int argc, char *argv[])
| {
|
|     /*-----*/
|     /* standard return codes          */
|     /*-----*/
|
| #define ERROR -1
| #define OK    0
|
|     /*-----*/
|     /* standard CCA parameters        */
|     /*-----*/
|
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 0;
|     char exit_data[2];
|     long rule_array_count = 0;
|     char rule_array[1][8];
|
|     /*-----*/
|     /* fields unique to this sample program          */
|     /*-----*/
|     char key_label[65] =
|         "*****";
|     long data_set_name_length = 0;
|     char data_set_name[65];
|     char security_server_name[9] = " ";
|
|     FILE *krl;
|     int keysto_type = PKA;
|     /*-----*/
|     /* Check whether the user requested to re-encipher a DES or */
|     /* a PKA keystore file. Default to PKA if key file type is */
|     /* not specified.          */
|     /*-----*/
|     if (argc >= 2)
|     {
| if ((strcmp(argv[1], "DES")==0))
| {

```

```

|     printf("%nDES ");
|     keysto_type = DES;
| }
| else if ((strcmp(argv[1],"PKA")==0))
|     printf("%nPKA ");
| else
| {
|     printf("%nKeystore type parm incorrectly specified.%n");
|     printf("Acceptable choices are PKA or DES.%n");
|     printf("The default is PKA.%n");
|     return ERROR;
| }
| }
| else
| {
| printf("%nPKA ");
| }
|
|     if (keysto_type == DES)
|     {
|
| /*-----*/
| /* Invoke the verb to create a DES Key Record List          */
| /*-----*/
| CSNBKRL( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         key_label,
|         &data_set_name_length,
|         data_set_name,
|         security_server_name);
|     }
|     else
|     {
| /*-----*/
| /* Invoke the verb to create a PKA Key Record List          */
| /*-----*/
| CSNDKRL( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (char *) rule_array,
|         key_label,
|         &data_set_name_length,
|         data_set_name,
|         security_server_name);
|     }
|
|     if ((return_code != 0) || (reason_code != 0))
|     {
| printf("Key Record List generation was unsuccessful. ");
| printf("Return/reason code = %d/%d\n",return_code, reason_code);
|     }
|     else
|     {
| printf("Key Record List generation was successful. ");
| printf("Return/reason codes = %d/%d\n",return_code, reason_code);
| data_set_name[data_set_name_length] = '\0';
| printf("data_set_name = %s\n",data_set_name);
|
| /* Open the Key Record List file. */
| kr1 = fopen(data_set_name, "rb");
|
| if (kr1 == NULL) /* Open failed. */

```

```

| {
|     printf("The open of the Key Record List file failed\n");
|     return ERROR;
| }
| else /* Open was successful. */
| {
|     char header1[77];
|     int num_rec, i;
|     long rec_length, offset_rec1;
|
|     /* Read the first part of the KRL header. */
|     fread(header1,1,77,kr1);
|
|     /* Get the number of key records in the file. */
|     num_rec = atoi(&header1[50]);
|     printf("Number of key records = %d\n",num_rec);
|
|     /* Get the length for the key records. */
|     rec_length = atol(&header1[58]);
|
|     /* Get the offset for the first key record. */
|     offset_rec1 = atol(&header1[62]);
|
|     /* Set the file pointer to the first key record. */
|     fseek(kr1, offset_rec1, SEEK_SET);
|
|     /* Loop through the entries in the KRL and re-encipher. */
|     for (i = 1; i <= num_rec; i++)
|     {
|         int result;
|         result = re_encipher(kr1, rec_length, keysto_type);
|         if (result !=0)
|         {
|             fclose(kr1);
|             return ERROR;
|         }
|         printf("Key store file re-enciphered successfully.\n\n");
|         fclose(kr1);
|         return OK;
|     }
| }
| }
|
| } /* end of main() */
|
| int re_encipher(FILE *key_rec, long rec_length, int key_type)
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|
|     long return_code;
|     long reason_code;
|     long exit_data_length = 0;
|     char exit_data[2];
|     long rule_array_count = 1;
|     char rule_array[1][8];
|
|     /*-----*/
|     /* fields unique to this function */
|     /*-----*/
|     long key_identifier_length = 64;
|     char key_identifier[64];
|     char key_record[154];

```

```

|     fread(key_record, 1, rec_length, key_rec);
|     memcpy(key_identifier, &key_record[3], 64);
|     memcpy(rule_array, "RTCMK ",8);
|
|     if (key_type == DES)
|     {
| CSNBKTC(&return_code,
| &reason_code,
| &exit_data_length,
| exit_data,
| &rule_array_count,
| (char *) rule_array,
| key_identifier);
|     }
|     else if (key_type == PKA)
|     {
| CSNDKTC(&return_code,
| &reason_code,
| &exit_data_length,
| exit_data,
| &rule_array_count,
| (char *) rule_array,
| &key_identifier_length,
| key_identifier);
|     }
|     else
|     {
| printf("re_encipher() called with an invalid key type.¥n");
| return ERROR;
|     }
|
|     printf("Re-enciphering for key_label = %.64s",key_identifier);
|     printf("completed with return/reason codes of ");
|     printf("%d/%d¥n",return_code,reason_code);
|     return return_code;
|
| }/* end of re_encipher() */

```

関連概念

87 ページの『鍵の再暗号化』

DCM および SSL で使用するための暗号化コプロセッサの構成

暗号化コプロセッサを SSL で使用できるようにする場合は、ここに記載された情報を参照してください。

以下のセクションでは、暗号化コプロセッサを SSL で使用するために必要なステップをリストしています。

DCM および SSL でコプロセッサを使用する

暗号化コプロセッサとその前提条件となるソフトウェアをインストールするには、以下のことを実行する必要があります。

- サーバーにコプロセッサをインストールします。

フィーチャー 4801 の場合は、暗号化コプロセッサに付属の 4801 PCI 暗号化コプロセッサ・カードの説明書に従って、暗号化コプロセッサをインストールします。

- i5/OS オプション 35 CCA CSP をインストールします。
- セキュア・アクセスを実現するための i5/OS オブジェクト権限を設定します。
- Web ブラウザーを使用して、<http://server-name:2001> の「System Tasks」ページにアクセスします。

- コプロセッサを構成します。
- 以上で、暗号化コプロセッサを使用して SSL 証明書用の秘密鍵を作成する準備が整いました。
- DCM を使用して証明書を作成します。このとき、秘密鍵をハードウェアで生成することを指定します。
 - DCM を使用して、署名された証明書を受信します。

注: SSL に複数のカードを使用する場合は、184 ページの『複数の暗号化コプロセッサの管理』と 195 ページの『マスター鍵の複製』を参照してください。

関連概念

- 184 ページの『複数の暗号化コプロセッサの管理』
区画ごとに最大 8 つの暗号化コプロセッサを持つことができます。サーバーごとにサポートされる暗号化コプロセッサの最大数は、システム・モードによって異なります。複数のコプロセッサを SSL で使用する場合は、このトピックを参照してください。
- 18 ページの『セキュア・アクセス』
アクセス制御は、システム資源の使用を、資源との対話を許可されているユーザーのみに制限します。サーバーでは、システム資源へのユーザーの権限を管理することができます。
- 21 ページの『暗号化コプロセッサの構成』
暗号化コプロセッサを構成すると、その暗号操作をすべて使用できるようになります。

i5/OS アプリケーションで使用するための暗号化コプロセッサの構成

このトピックでは、暗号化コプロセッサを i5/OS アプリケーションで使用できるようにするために必要なステップをリストしています。

i5/OS アプリケーションでの暗号化コプロセッサの使用

暗号化コプロセッサとその前提条件となるソフトウェアをインストールするには、以下のことを実行する必要があります。

- サーバーにコプロセッサをインストールします。
- フィーチャー 4801 の場合は、暗号化コプロセッサに付属の 4801 PCI 暗号化コプロセッサ・カードの説明書に従って、暗号化コプロセッサをインストールします。
- i5/OS オプション 35 CCA CSP をインストールします。
- セキュア・アクセスを実現するための i5/OS オブジェクト権限を設定します。
- Web ブラウザーを使用して、<http://server-name:2001> の「System Tasks」ページにアクセスします。
- コプロセッサを構成します。
- 暗号化コプロセッサを使用するアプリケーションを作成します。

注: i5/OS アプリケーションに複数のカードを使用する場合は、184 ページの『複数の暗号化コプロセッサの管理』を参照してください。

暗号化コプロセッサへの移行

以前に暗号化を使用したことがある場合には、以前の暗号化製品から 4764 または 4758 暗号化コプロセッサにマイグレーションするための要件が発生する場合があります。

注: 4758 から 4764 への移行:

4758 暗号化コプロセッサを新しい 4764 暗号化コプロセッサで置き換える場合には、4764 コプロセッサの役割およびプロファイルを、4758 暗号化コプロセッサで使用していたものと同じになるようにセットアップしてください。4758 および 4764 暗号化コプロセッサは、同じ CCA API および鍵ストア・ファイルを使用できます。

Cryptographic Support for OS/400® (5722-CR1) の暗号クロスドメイン・ファイルがある可能性があります。または、IBM CCA Services for OS/400 (5799-FRF) 製品の鍵ストア・ファイルがある可能性があります。いずれかのファイルが存在する場合には、その内容を新規の暗号化コプロセッサに移行することができます。以下に、各暗号製品に使用可能な移行プログラムの例をあげます。

- **Cryptographic Support for AS/400® または i5/OS (5769-CR1 または 5722-CR1):** Cryptographic Support は、ホスト・マスター鍵の下にあるクロスドメイン鍵を暗号化するソフトウェア専用の製品です。また、Cryptographic Support は、クロスドメイン鍵をファイルに保管します。クロスドメイン鍵ファイルは、Cryptographic Support for AS/400 または i5/OS から暗号化コプロセッサに移行することができます。『Cryptographic Support for AS/400 のクロスドメイン鍵ファイルの移行』を参照してください。
- **IBM CCA サービス (5799-FRF) PRPQ:** この製品では、データ暗号化規格 (DES) を使用することにより暗号化ハードウェア上で暗号機能を提供します。CCA サービス PRPQ を使用するには、暗号プロセッサ、フィーチャー番号 2620 または 2628 がサーバーに導入されている必要があります。鍵ストア・ファイルは、IBM CCA サービスから暗号化コプロセッサに移行することができます。『IBM CCA Services for OS/400 PRPQ からの鍵ストア・ファイルの移行』を参照してください。

IBM CCA Services for OS/400 PRPQ からの鍵ストアファイルの移行

現在、Common Cryptographic Architecture (CCA) Services for OS/400 (5799-FRF) を使用している場合には、暗号化コプロセッサが使用できるように、鍵ストア・ファイル内にある鍵を移行することができます。コプロセッサは、移行された鍵を CCA Cryptographic Service Provider (i5/OS オプション 35 としてパッケージされている CCA CSP) と共に使用します。

注: CCA サービスは暗号化コプロセッサよりも広範囲の鍵タイプをサポートしているため、すべての鍵を移行することはできません。例えば、制御ベクトルに禁止エクスポート・ビットを設定している鍵を移行することはできません。さらに、CCA サービスには暗号化コプロセッサとは大幅に異なる公開鍵アルゴリズム (PKA) を提供しているため、CCA Services では、いかなる PKA 鍵も移行することができません。

データ暗号化規格 (DES) 鍵を移行するには、2 つのプログラムを作成する必要があります。CCA は、外部 DES 鍵トークンの形式を定義しているため、CCA は両製品に対して同じです。あるいは、2 つのプログラム例『例: 鍵のエクスポート』および『例: 鍵のインポート』を変更して、鍵ストア・ファイルを移行することができます。CCA は、外部 DES 鍵トークンの形式を定義しているため、CCA は両製品に対して同じです。

EXPORT プログラムを IMPORT プログラムと一緒に使用します。これにより、DES 鍵は、IBM CCA Services から暗号化コプロセッサおよび CCA CSP に移行されます。最初に EXPORT プログラムを実行して、必要な鍵情報が安全でエクスポート可能な形式で組み込まれているファイルを生成します。次に、ファイルをターゲット・サーバーに転送します。IMPORT プログラムを実行すると、作成した鍵ストア・ファイルに鍵をファイルからインポートすることができます。鍵をインポートする鍵ストア・ファイルは、プログラムを実行する前にすでに存在していなければなりません。

注: 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

- | プログラム例を変更するには、次のステップを実行します。
- | 1. 鍵暗号化鍵に対して同じクリア鍵値を両方の製品にインポートします。鍵暗号化鍵は、CCA サービス
| に対しては EXPORTER になり、CCA CSP に対しては IMPORTER になります。
- | 2. 移行したい各鍵ごとに CCA サービスの Key_Export (CSNBKEX) CCA API を実行します。実行する
| と、プログラム例は API を呼び出します。
- | 3. Key_Import (CSNBKIM) CCA API を使用して、出力された外部鍵トークンを CCA CSP および暗号化
| コプロセッサにインポートします。各鍵ごとにインポートするためにはプログラムを変更することを
| 忘れないでください。

| 各鍵を処理するためにプログラムをいったん変更すると、そのプログラムを実行することができます。最初
| に EXPORT を実行してから IMPORT を必ず実行してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| **例: 鍵のエクスポート:**

| この鍵ストア・ファイルを移行するには、必要に応じて以下のプログラム例を変更してください。

| 以下は、ステップ 1 です。このプログラムを実行した後 106 ページの『例: 鍵のインポート』を使用して
| 移行プロセスを完了します。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
| ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのはではなく、これ
| らのプログラム例を修正して使用することをお勧めします。

```
| /*-----*/
| /* Description:  One of two programs used to migrate DES keys   */
| /*                from a key store file used with the 2620 to a  */
| /*                key store file for use with the card.          */
| /*                */
| /* Note: This program is intended to be used in conjunction with */
| /*      IMPORT_TSS to migrate DES keys from 2620 to card.        */
| /*      EXPORT_TSS should be run first to generate a file       */
| /*      containing the needed key information in a secure,      */
| /*      exportable form. The file should then be transferred   */
| /*      to the target system. IMPORT_TSS can then be run using  */
| /*      the file to import the keys into a previously created   */
| /*      key storage file.                                       */
| /*                */
| /*                */
| /* COPYRIGHT      5769-SS1 (c) IBM Corp 1999                  */
| /*                */
| /* This material contains programming source code for your     */
| /* consideration. These examples have not been thoroughly     */
| /* tested under all conditions. IBM, therefore, cannot        */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these programs. All programs contained herein are       */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF        */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  */
| /* EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files.                                   */
| /*                */
| /* Parameters: File to contain exported key information        */
| /*                */
| /* Examples:                                                  */
```

```

| /* CALL PGM(EXPORT_TSS) PARM('File_for_Exported_Keys') */
| /* */
| /* */
| /* Use the following commands to compile this program: */
| /* ADDLIB LIB(QTSS) */
| /* CRTCMOD MODULE(EXPORT_TSS) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(EXPORT_TSS) MODULE(EXPORT_TSS) */
| /* */
| /* Note: authority to the functions CSNBKEX, CSNBKPI, CSNBKRL, */
| /* and CSNBKTB is assumed */
| /* */
| /* Common Cryptographic Architecture (CCA) verbs used: */
| /* Key_Export CSNBKEX */
| /* Key_Part_Import CSNBKPI */
| /* Key_Record_List CSNBKRL */
| /* Key_Token_Build CSNBKTB */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include "MIPTRNAM.H" /* needed to resolve function ptrs */
| #include "csucinl.h" /* header file for CCA Cryptographic
| Service Provider */
|
| int main(int argc, char *argv[])
| {
|
| /*-----*/
| /* standard return codes */
| /*-----*/
| #define ERROR -1
| #define OK 0
|
| /*-----*/
| /* Declare function pointers (see csucinl.h) */
| /*-----*/
| T_CSNBKEX *CSNBKEX;
| T_CSNBKRL *CSNBKRL;
| T_CSNBKPI *CSNBKPI;
| T_CSNBKTB *CSNBKTB;
|
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code;
| long reason_code;
| long exit_data_length = 0;
| char exit_data[2];
| long rule_array_count = 0;
| char rule_array[2][8];
|
| /*-----*/
| /* additional parameters needed for CSNBKRL */
| /*-----*/
| char key_label[64];
| long data_set_name_length = 0;
| char data_set_name[65];
| char security_server_name[9] = " ";
|
| /*-----*/
| /* additional parameters needed for CSNBKEX */
| /*-----*/
| char key_type[8];

```

```

char source_key_identifier[64];
char exporter_key_identifier[64];
char target_key_token[64];

/*-----*/
/* additional parameters needed for CSNBKTB */
/*-----*/
char key_token[64];
char key_value[64];
long master_key_verification_pattern = 0;
long reserved_int;
char reserved_str[8];
char control_vector[16];

/*-----*/
/* additional parameters needed for CSNBKPI */
/*-----*/
char key_part[16];
char key_identifier[64];

/*-----*/
/* Other variables */
/*-----*/
char header1[77];
long num_rec, i;
long num_successful = 0;
long rec_length, offset_rec1;

char key_record[154];

FILE *kr1, *export_file;

/* Check input parm */
if (argc < 2)
{
    printf("File for storing the exported key data not specified.\n");
    return ERROR;
}

/*-----*/
/* Resolve function pointers */
/*-----*/
_lib_qualify(CSNBKEX,QTSS)
_lib_qualify(CSNBKRL,QTSS)
_lib_qualify(CSNBKPI,QTSS)
_lib_qualify(CSNBKTB,QTSS)

memset(key_label, ' ',64);
memcpy(key_label, "*****",9);

/*-----*/
/* Invoke the verb to create a DES Key Record List */
/*-----*/
CSNBKRL( &return_code,
         &reason_code,
         &exit_data_length,
         exit_data,
         key_label,
         &data_set_name_length,
         data_set_name,
         security_server_name);

if ((return_code != 0) || (reason_code != 0))

```

```

| {
|     printf("Key Record List generation was unsuccessful. ");
|     printf("Return/reason code = %d/%d\n",return_code, reason_code);
|     return ERROR;
| }
|
| printf("Key Record List generation was successful. ");
| printf("Return/reason codes = %d/%d\n",return_code, reason_code);
| data_set_name[data_set_name_length] = '\0';
| printf("data_set_name = %s\n",data_set_name);
|
| /* Generate a clear key for export use. */
| /* The same key will be used for import. */
| memcpy(key_type,"EXPORTER",8);
| rule_array_count = 2;
| memcpy(rule_array[0],"INTERNAL",8);
| memcpy(rule_array[1],"KEY-PART",8);
|
| CSNBKTB( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         key_token,
|         key_type,
|         &rule_array_count,
|         (char *) rule_array,
|         key_value,
|         &master_key_verification_pattern,
|         &reserved_int,
|         reserved_str,
|         control_vector,
|         reserved_str,
|         &reserved_int,
|         reserved_str,
|         reserved_str);
|
| if (return_code != 0) {
|     printf("Building of the export key failed.\n");
|     printf("Key Token Build failed.");
|     printf("Return/reason codes = %d/%d\n",return_code, reason_code);
|     return ERROR;
| }
|
| /* Import the key parts to be used. */
| rule_array_count = 1;
| memcpy(rule_array[0],"FIRST  ",8);
| memset(key_part,'\0',16);
|
| for(i=1;i<=2;i++) {
|     CSNBKPI( &return_code,
|             &reason_code,
|             &exit_data_length,
|             (char *) exit_data,
|             &rule_array_count,
|             (char *) rule_array,
|             key_part,
|             key_token);
|
|     if (return_code != 0) {
|         printf("Building of the export key failed.\n");
|         printf("Key Part Import failed.");
|         printf("Return/reason codes = %d/%d\n",return_code, reason_code);
|         return ERROR;
|     }
| }

```

```

|     memcpy(rule_array[0],"LAST  ",8);
|     /* Set key_part to the clear key to be used. */
|     /* Note: It may not be desirable to hard-code this. */
|     memcpy(key_part,"CLEAR.KEY.hErE!!",16);
|
| }
|
| /* Export key built successfully. */
| /* Open the Key Record List file. */
| krl = fopen(data_set_name, "rb");
|
| if (krl == NULL)
| { /* Open failed. */
|     printf("The open of the Key Record List file failed.\n");
|     return ERROR;
| }
|
| /* Key record list open was successful. */
| /* Open the file to save key info. */
| export_file = fopen(argv[1], "wb");
| if (export_file == NULL)
| {
|     printf("Opening of key export file failed.\n");
|     fclose(krl);
|     return ERROR;
| }
|
| /* Write num_successful to the export file to hold a place for it. */
| fwrite(&num_successful,sizeof(long),1,export_file);
|
| /* Read the first part of the KRL header. */
| fread(header1,1,77,krl);
|
| /* Get the number of key records in the file. */
| num_rec = atoi(&header1[50]);
| printf("Number of key records = %d\n",num_rec);
|
| /* Get the length for the key records. */
| rec_length = atol(&header1[58]);
|
| /* Get the offset for the first key record. */
| offset_rec1 = atol(&header1[62]);
|
| /* Set the file pointer to the first key record. */
| fseek(krl, offset_rec1, SEEK_SET);
|
| /* Set the key type to TOKEN. */
| memcpy(key_type,"TOKEN  ",8);
|
| /* Loop through the entries in the KRL and EXPORT. */
| for (i = 1; i <= num_rec; i++)
| {
|     fread(key_record, 1, rec_length, krl);
|     memcpy(source_key_identifier, &key_record[3], 64);
|
|     CSNBKEX(&return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             key_type,
|             source_key_identifier,
|             key_token,
|             /* exporter_key_identifier, */
|             target_key_token);
| }

```

```

|     printf("Exporting of key = %.64s",source_key_identifier);
|     printf("completed with return/reason codes of ");
|     printf("%d/%d\n",return_code,reason_code);
|
|     if (return_code == 0)
|     {
|         ++num_successful;
|         fwrite(source_key_identifier, 1, 64, export_file);
|         fwrite(target_key_token, 1, 64, export_file);
|     }
|
| } /* end of for loop */
|
| printf("Key store file exported successfully.\n");
| printf("%d key(s) successfully exported.\n\n",num_successful);
|
| /* Write out the number of exported keys and close the file. */
| fseek(export_file,0,SEEK_SET);
| fwrite(&num_successful,sizeof(long),1,export_file);
|
| /* Close the files and return. */
| fclose(kr1);
| fclose(export_file);
| return OK;
| }

```

| 例: 鍵のインポート:

| 必要に応じてこのプログラム例を変更して、鍵ストア・ファイルの移行を完了してください。

| 以下は、ステップ 2 です。これまでに実行していない場合には、101 ページの『例: 鍵のエクスポート』プログラムを実行して移行プロセスを開始します。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| /*-----*/
| /* Description:  One of two programs used to migrate DES keys   */
| /*               from a key store file used with the 2620 to a  */
| /*               key store file for use with the card.         */
| /*               */
| /* Note: This program is intended to be used in conjunction with */
| /*       EXPORT_TSS to migrate DES keys from 2620.             */
| /*       EXPORT_TSS should be run first to generate a file     */
| /*       containing the needed key information in a secure,    */
| /*       exportable form. The file should then be transferred  */
| /*       to the target system. IMPORT_TSS can then be run using */
| /*       the file to import the keys into a previously created  */
| /*       key storage file.                                     */
| /*               */
| /*               */
| /* COPYRIGHT      5769-SS1 (c) IBM Corp 1999                 */
| /*               */
| /* This material contains programming source code for your     */
| /* consideration. These examples have not been thoroughly     */
| /* tested under all conditions. IBM, therefore, cannot       */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these programs. All programs contained herein are      */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF        */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  */

```



```

| /* EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* Parameters: File containing exported key information */
| /* */
| /* Examples: */
| /* CALL PGM(IMPORT_TSS) PARM('Exported_Key_File') */
| /* */
| /* Note: The CCA verbs used in the this program are more fully */
| /* described in the IBM CCA Basic Services Reference */
| /* and Guide (SC31-8609) publication. */
| /* */
| /* Note: This program assumes the card you want to use is */
| /* already identified either by defaulting to the CRP01 */
| /* device or has been explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* This program also assumes the key store file you will */
| /* use is already identified either by being specified on */
| /* the cryptographic device or has been explicitly named */
| /* using the Key_Store_Designate verb. Also you must be */
| /* authorized to update records in this file. */
| /* */
| /* Use the following commands to compile this program: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(IMPORT_TSS) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(IMPORT_TSS) MODULE(IMPORT_TSS) */
| /* BNDSRVPGM(QCCA/CSNBKRC QCCA/CSNBKIM QCCA/CSNBKPI) */
| /* */
| /* Note: authority to the CSNBKIM, CSNBKPI, and CSNBKRC */
| /* service programs in the QCCA library is assumed. */
| /* */
| /* Common Cryptographic Architecture (CCA) verbs used: */
| /* Key_Import CSNBKIM */
| /* Key_Record_Create CSNBKRC */
| /* Key_Part_Import CSNBKPI */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h" /* header file for CCA Cryptographic
| Service Provider */
|
| /*-----*/
| /* Structure defining the DES key token for internal keys. This */
| /* structure is used in the creation of the importer key- */
| /* encrypting key. For more information on the fields in this */
| /* structure, see the IBM CCA Basic Services Reference and */
| /* Guide (SC31-8609-01), Appendix B and C. */
| /*-----*/
| struct DES_key_token {
|     char type; /* Set to 0x01 for 'internal' */
|     char resv1; /* Reserved (set to binary zero) */
|     char mkvp[2]; /* Master Key Verification Pattern */
|     char version; /* Version. Will be set to 0x03. */
|     char resv2; /* Reserved (set to binary zero) */
|     char flag; /* Flag */
|     char resv3; /* Reserved (set to binary zero) */
|     char resv4[8]; /* Reserved (set to binary zero) */
|     char key1[8]; /* Single length encrypted key or
| left half of double length
| encrypted key. */
|     char key2[8]; /* Null or right half of double
| length encrypted key */
|

```

```

|     int     cvb1[2];      /* Control-vector base          */
|     int     cvb2[2];      /* Null or control vector base for
|                          the 2nd eight-byte portion of a
|                          16-byte key          */
|     char    resv5[12];    /* Reserved (set to binary zero) */
|     int     tvv;         /* Token-validation value        */
| };
|
| int main(int argc, char *argv[])
| {
|
|     /*-----*/
|     /* standard return codes          */
|     /*-----*/
|
| #define ERROR -1
| #define OK    0
|
|     /*-----*/
|     /* standard CCA parameters        */
|     /*-----*/
|
|     long return_code;
|     long reason_code;
|     long exit_data_length = 0;
|     char exit_data[2];
|     long rule_array_count = 0;
|     char rule_array[2][8];
|
|     /*-----*/
|     /* additional parameters required for CSNBKRC and CSNBKIM */
|     /*-----*/
|     char import_key_label[64];
|     char import_key_token[64];
|
|     /*-----*/
|     /* additional parameters required for CSNBKPI          */
|     /*-----*/
|     struct DES_key_token importer_kt;
|
|     char importer_key_token[64];
|     char key_type[8];
|     char key_part[16];
|
|     /*-----*/
|     /* Other variables          */
|     /*-----*/
|     long num_rec = 0, i;
|     long num_imported = 0;
|
|     FILE *import_file;
|
|     printf("%n%n");
|     /* Check input parm */
|     if (argc < 2)
|     {
|         printf("File containing the exported key data not specified.%n");
|         return ERROR;
|     }
|
|     /* Generate a clear key for import use. */
|
|     /* Initialize the importer key token. */
|     memset(&importer_kt,0x00,sizeof(struct DES_key_token));
|     importer_kt.type = 0x01;

```

```

importer_kt.version = 0x03;
importer_kt.flag = 0x40; /* Indicates control vector is present */
importer_kt.cvb1[0] = 0x00427d00;
importer_kt.cvb1[1] = 0x03480000;
importer_kt.cvb2[0] = 0x00427d00;
importer_kt.cvb2[1] = 0x03280000;
importer_kt.tvv = 0x0af53a00;

/* Initialize parameters for the first pass */
rule_array_count = 1;
memcpy(rule_array[0], "FIRST ", 8);
memset(key_part, 0x01, 16);

for(i=1; i<=2; i++) {
    CSNBKPI( &return_code,
            &reason_code,
            &exit_data_length,
            (char *) exit_data,
            &rule_array_count,
            (char *) rule_array,
            key_part,
            (char *) &importer_kt);

    if (return_code != 0) {
        printf("Building of the importer key failed.\n");
        printf("Key Part Import failed.");
        printf("Return/reason codes = %d/%d\n", return_code, reason_code);
        return ERROR;
    }
    else if ( i == 1) {
        /* Init variables for the final pass */
        memcpy(rule_array[0], "LAST ", 8);
        /* Set key part to the clear key to be used. */
        memcpy(key_part, "Clear.KEY.hErE!!", 16);
    }
}

/* Import key built successfully. */
printf("Importer key built successfully.\n\n");

/* Open the Exported Key file. */
import_file = fopen(argv[1], "rb");

if (import_file == NULL)
{ /* Open failed. */
    printf("The open of the Exported Key file failed\n");
    return ERROR;
}

/* Import Key file open was successful. */
fread(&num_rec, sizeof(num_rec), 1, import_file);

/* Loop through the entries in the import file and create key records. */
for (i = 1; i <= num_rec; i++)
{
    fread(import_key_label, 1, 64, import_file);
    fread(import_key_token, 1, 64, import_file);

    printf("Importing DES key:\n");
    printf("    %%.64s%\n", import_key_label);

    /* Create a key record. */
    CSNBKRC(&return_code,
            &reason_code,
            &exit_data_length,

```

```

        exit_data,
        import_key_label);

    if (return_code != 0)
    {
        printf("    Key record creation failed. ");
        printf("Return/reason codes = %d/%d\n\n",return_code,reason_code);
        continue;
    }

    /* Else, key record created successfully so import the key. */
    memcpy(key_type,"TOKEN  ",8);

    CSNBKIM( &return_code,
            &reason_code,
            &exit_data_length,
            exit_data,
            key_type,
            import_key_token,
            (char *) &importer_kt,
            import_key_label);

    if (return_code != 0)
    {
        printf("    Key import failed. ");
        printf("Return/reason codes = %d/%d\n\n",return_code,reason_code);
        continue;
    }

    /* else, Key import was a success. */
    printf("    Key imported successfully. ");
    printf("Return/reason codes = %d/%d\n\n",return_code,reason_code);
    ++num_imported;
} /* end of for loop */

printf("\nCompleted key import procedure.\n");
printf("%d of %d key(s) successfully imported.\n\n",num_imported,num_rec);
fclose(import_file);
return OK;
}

```

Cryptographic Support のシステム・クロスドメイン・キー・ファイルの移行

以前にサーバーで暗号の作業を行っている場合には、Cryptographic Support (5769-CR1) の暗号クロスドメイン・ファイルがある可能性があります。既存のクロスドメイン鍵は、新しい暗号化コプロセッサに移行することができます。

Cryptographic Support for OS/400 製品 (5769-CR1 または 5722-CR1) は、ホスト・マスター鍵で自身のクロスドメイン鍵を暗号化して、ファイルに保管します。Common Cryptographic Architecture (CCA) は、この形式では使用できませんが、コプロセッサと共に CCA を使用するために、クロスドメイン鍵を Cryptographic Support 製品から移行することができます。この作業を完了する前に、多くの点に考慮する必要があります。

- **クロスドメイン鍵によるクロスドメイン鍵の暗号化:** Cryptographic Support は、クロスドメイン鍵のクリア鍵値のインポートをサポートし、さらにクロスドメイン鍵の下でデータ鍵の暗号化をサポートします。ただし、クロスドメイン鍵の下でのクロスドメイン鍵の暗号化はサポートしておらず、クロスドメイン鍵のクリア鍵値を戻すこともサポートしていません。したがって、クロスドメイン鍵の移行は、単にエクスポート操作やインポート操作を実行するよりもかなり複雑になります。
- **単一長鍵と倍長鍵の比較:** Cryptographic Support のすべての鍵は、単一長鍵です。CCA では、すべての鍵暗号化鍵および PIN 鍵が倍長鍵です。鍵の長さは異なりますが、単一長鍵から倍長鍵を作成し、その

倍長鍵を単一長鍵のように振る舞わせることができます。倍長鍵の両半分が同じ場合、暗号操作の結果は単一長鍵を使用した場合と同じになります。したがって、鍵を Cryptographic Support から CCA に移行する場合は、クロスドメイン鍵の鍵値を、CCA 鍵の鍵値フィールドの両半分にコピーする必要があります。

- **CCA 制御ベクトルとマスター鍵変形の比較:**CCA では、鍵暗号鍵で鍵が暗号化されている、といわれる場合は、実際には、鍵暗号化鍵と制御ベクトルの排他論理和演算で得られる鍵で暗号化されています。暗号化機能サポートの場合は、クロスドメイン鍵は、3 つの異なるマスター鍵変形の 1 つで暗号化されます。マスター鍵変形とは、ホスト・マスター鍵と、16 進値 22、44、または 88 の上位または下位のどちらかの 8 バイトとの排他論理和演算の結果です。制御ベクトルとマスター鍵変形はどちらも、鍵の分離を提供しているため、鍵をそれらの本来の使用に制限しています。CCA では、制御ベクトルの値により、その使用方法が決まります。暗号化機能サポートでは、鍵の使い方により、その鍵を暗号化解除するために使用されるマスター鍵変形が決まります。どちらの場合も、鍵をその意図している使い方以外で使用しようとする、エラーが発生します。制御ベクトルとマスター鍵変形の動作は似ていますが、マスター鍵変形を形成するために使用される値は、制御ベクトルと同じではありません。

- **倍長鍵の CCA 制御ベクトルの非対称性:** 倍長鍵は、その両半分が同じ場合のみ、単一長鍵のように動作します。倍長鍵の制御ベクトルは非対称です。制御ベクトルと排他論理和演算された倍長鍵は、いかなる場合でも、生成された鍵の両半分は同じにはなりません。この倍長鍵は、単一長鍵のように動作しません。

鍵を移行するためには 2 つの方法のうちの 1 つを選択することができます。

鍵を移行するためには 2 つの方法のうちの 1 つを選択することができます。

方法 1 (推奨)

この方法は、上に示した考慮事項に対するいくつかの解決策を示しています。この方法を使用することをお勧めします。

クロスドメイン鍵を暗号化機能サポートから CCA に移行するには、両方に共通の鍵暗号化鍵を使用する必要があります。暗号化機能サポートのホスト・マスター鍵を、暗号化機能サポートと CCA の間の共通鍵として使用することができます (CCA では、ホスト・マスター鍵は、マスター鍵として知られていません)。暗号化機能サポートのホスト・マスター鍵のクリア値を CCA に IMPORTER 鍵暗号化鍵としてインポートします。ホスト・マスター鍵を 2 つのパーツとして入力するため、CCA へのホスト・マスター鍵のインポートは、Key_Part_Import (CSNBKPI) CCA API を使用して、2 つのパーツとしてインポートすると思えばなりません。Cryptographic Support のホスト・マスター鍵に対して二重の責務がある場合、この鍵暗号化鍵に対してもこの二重の責務を維持する必要があります。代わりに、ホスト・マスター鍵の両方のパーツが分かっている場合には、2 つのパーツの排他的論理和を実行して鍵を 1 つのパーツとしてインポートすることもできます。プログラム例では、この方法を使用してホスト・マスター鍵をインポートしています。プログラム例で行っているように、ホスト・マスター鍵をすべてのクロスドメイン鍵の移行と組み合わせる代わりに、完全に別個のプロセスでホスト・マスター鍵をインポートすることを考慮することもできます。

クロスドメイン鍵には次の 3 つのタイプがあります。

- 受信クロスドメイン鍵
- 送信クロスドメイン鍵
- PIN クロスドメイン鍵

受信クロスドメイン鍵は、CCA では IMPORTER 鍵暗号化鍵と呼ばれます。どちらも、暗号化された鍵の受信あるいはインポートに使用されます。

送信クロスドメイン鍵は、a) データ鍵の暗号化 (暗号化された鍵は、別のシステムに送信できます) と、b) 暗号化された個人識別番号 (PIN) の変換、に使用されます。CCA には暗号サポート製品よりも厳密な鍵分割があるため、両機能を実行する鍵を生成またはインポートすることはできません。EXPORTER 鍵暗号化鍵および OPINENC (アウトバウンド PIN 暗号化) 鍵の両方として鍵を使用する場合には、2 つの異なる鍵タイプを持つ 2 種類の鍵に送信クロスドメイン鍵を 2 回インポートする必要があります。

PIN クロスドメイン鍵は、PIN の生成と検証に使用することができます。CCA は、これらの 2 つの使い方を、PINGEN (PIN 生成) と PINVER (PIN 検証) 鍵に分離します。鍵を、PIN の生成と検証の両方に使用する場合は、PIN クロスドメイン鍵を同様に 2 回インポートする必要があります。

ホスト・マスター鍵がデータ鍵を暗号化するとき、異なるマスター鍵変形がクロスドメイン鍵を暗号化します。


- マスター鍵変形 1 は送信クロスドメイン鍵を暗号化する。バリエーション 1 は、16 進数 88 の 8 バイトでホスト・マスター鍵を排他論理和演算した結果です。

- マスター鍵変形 2 は受信クロスドメイン鍵を暗号化する。バリエーション 2 は、ホスト・マスター鍵および 16 進数 22 の 8 バイトを排他論理和演算した結果です。

- マスター鍵変形 3 は PIN クロスドメイン鍵を暗号化する。バリエーション 3 は、ホスト・マスター鍵および 16 進数 44 の 8 バイトを排他論理和演算した結果です。

注: ホスト・マスター鍵のクリア鍵値だけを CCA にインポートする場合には、どの鍵も移行することができません。鍵を移行するには、どのマスター鍵変形で鍵を暗号化するかを考慮する必要があります。

マスター鍵変形を作成するための 8 バイトの値は、制御ベクトルに類似しています。鍵の移行プロセスは、鍵の制御ベクトルを変更するプロセスとして考えることができます。このプロセスの方法については、「IBM PCI 暗号化コプロセッサ CCA 基本サービスのリファレンスおよびガイド (IBM PCI

Cryptographic Coprocessor CCA Basic Services Reference and Guide)」 を参照してください。この方法は、**事前排他論理和演算**技法です。鍵をインポートする前に鍵暗号化鍵 (この場合はホスト・マスター鍵) のクリア鍵値が制御ベクトル情報で排他論理和されている場合には、この鍵暗号化鍵がインポートするどの鍵に対しても制御ベクトルを効果的に変更することができます。

「事前排他論理和演算」技法は、単一長鍵で作業を行っている場合には正しく動作します。倍長鍵の場合には、CCA 鍵の右半分の制御ベクトルが左半分の制御ベクトルとは違っているため、この技法を変更しなければなりません。この違いに対処するため、次のように鍵を 2 回インポートします。

1. 半分の各 8 バイトが、インポートする鍵の制御ベクトルの左半分と完全に同じになるように 16 バイト値を作成します。この 16 バイト値を事前排他論理和演算技法で使用して、「左インポーター」として参照することができるインポーター鍵暗号化鍵を作成します。この鍵暗号化鍵を使用してインポートされる鍵の左半分のみが有効になります。

2. 半分の各 8 バイトが、インポートする鍵の制御ベクトルの右半分と完全に同じになるように別の 16 バイト値を作成します。この 16 バイト値を事前排他的論理和技法で使用して、インポーター鍵暗号化鍵を作成します。このインポーター鍵暗号化鍵を使用すると、インポートされる鍵の右半分のみが有効になります。

3. 次のように、クロスドメインを 2 回インポートします。

- a. 最初に、ステップ 1 で作成された鍵暗号化鍵を使用して、結果の左半分を保管します。

- b. 次にステップ 2 で作成された鍵暗号化鍵を使用して、結果の右半分を保管します。

4. 最後のステップでは、ステップ A の結果の左半分を、ステップ B の結果の右半分と連結し、結合された結果を新しい鍵トークンとして配置します。

これで、Cryptographic Support 製品からクロスドメイン鍵のように働く CCA 倍長鍵が作成されました。すべてのクロスドメイン鍵をインポートするために必要なすべてのインポーター鍵暗号鍵の要約、およびインポーター鍵暗号鍵の作成手順については、『インポーター鍵暗号鍵の使用』を参照してください。

関連タスク

125 ページの『インポーター鍵暗号鍵の使用』

すべてのクロスドメイン鍵をインポートするために必要な すべてのインポーター鍵暗号鍵の要約については、ここに記載された情報を参照してください。ここでは、そのインポーター鍵暗号鍵の作成方法も示しています。

方法 2

この方法は、ご使用のシステムと環境のセキュリティに関して信頼がおけるものと思われる場合にのみ使用してください。この方法は上述の推奨方法よりも簡単ですが、クロスドメイン鍵ファイルにとってはセキュリティ・リスクが大きくなります。クロスドメイン鍵はアプリケーション記憶域では、クリアな形式になっているからです。

1. `Clear_Key_Import (CSNBCKI)` CCA API を使用して、ホスト・マスター鍵をデータ鍵として CCA にインポートします。以下に示すように、この鍵を、マスター鍵変形に等価なデータ鍵を作成するために必要な値と排他論理和演算することを忘れないでください。
 - a. マスター鍵変形 1 は送信クロスドメイン鍵を暗号化する。バリエーション 1 は、16 進数 88 の 8 バイトでホスト・マスター鍵を排他論理和演算した結果です。
 - b. マスター鍵変形 2 は受信クロスドメイン鍵を暗号化する。バリエーション 2 は、ホスト・マスター鍵および 16 進数 22 の 8 バイトを排他論理和演算した結果です。
 - c. マスター鍵変形 3 は PIN クロスドメイン鍵を暗号化する。バリエーション 3 は、ホスト・マスター鍵および 16 進数 44 の 8 バイトを排他論理和演算した結果です。

このステップを完了した後、3 つの異なるデータ鍵が作成されます。

2. `Decrypt (CSNBDEC)` CCA API を使用してクロスドメイン鍵を暗号化解除し、クリア鍵値を戻します。正しいデータ鍵を使用して暗号化を解除してください。

3. `Key_Part_Import (CSNBKPI)` CCA API を使用してクリア鍵を CCA にインポートします。

この方法は安全であるとは考えないでください。この方法の実行中、いずれかの時点で、すべての鍵はアプリケーション記憶域でクリアな形式になります。

完了です。以上で、クロスドメイン鍵をマイグレーションするプログラムを作成することも、Cryptographic Support のクロスドメイン鍵ファイルをご使用の暗号化コプロセッサにマイグレーションするための要件に合わせて、以下のプログラム例を変更することもできます。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```
l /*****  
l /* This program migrates keys stored in the file QACRKTBL in library */  
l /* QUSRSYS to key storage for Option 35 - CCA Cryptographic Service */  
l /* Provider. The QACRKTBL file contains cross domain keys that are */  
l /* used for the Cryptographic Support licensed program, 5722-CR1. */  
l /* */  
l /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999 */  
l /* */  
l /* This material contains programming source code for your */  
l /* consideration. These examples have not been thoroughly */  
l /* tested under all conditions. IBM, therefore, cannot */  
l /* guarantee or imply reliability, serviceability, or function */  
l /* of these program. All programs contained herein are */
```

```

| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* */
| /* The keys are migrated by the following steps: */
| /* */
| /* 1 - The master key used for 5722-CR1 passed as a parameter. */
| /* 2 - Build importer keys using the master key, 8 bytes of a mask */
| /* to create a variant, and a control vector. */
| /* 3 - The file QACRKTBL is opened for input. */
| /* 4 - A record is read. */
| /* 5 - Import the key using the pre-exclusive OR process. CCA uses */
| /* control vectors while non-CCA implementations don't. 5722-CR1*/
| /* creates master key variants similar to what 4700 finance */
| /* controllers do. Since the control vector and master key */
| /* variant material affect how the key is enciphered, the pre- */
| /* exclusive OR process "fixes" the importer key so that it can */
| /* correctly import a key. */
| /* - *SND keys are imported twice as an EXPORTER and OPINENC keys. */
| /* - *PIN keys are imported twice as a PINGEN and IPINENC keys. */
| /* - *RCV keys are imported as a IMPORTER key. */
| /* 6- A key record is created with a similar name as in QACRKTBL. */
| /* For key names longer than 8 characters, a '.' will be */
| /* inserted between the 8th and 9th characters. Also a 1 byte */
| /* extension is appended that describes the key type. */
| /* For example, MYKEY *RCV ----> MYKEY.R */
| /* MYKEK00001 *RCV ----> MYKEK000.01.R */
| /* */
| /* For *SND and *PIN keys, a second key record is also created. */
| /* For example, MYKEY *SND ----> MYKEY.S */
| /* MYKEY.O */
| /* MYPINKEY *PIN ----> MYPINKEY.P */
| /* MYPINKEY.I */
| /* */
| /* 7 - The key is written out to key store. */
| /* */
| /* 8 - Steps 4 through 7 are repeated until all keys have been */
| /* migrated. */
| /* */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* nonCCA master key - 8 bytes */
| /* */
| /* Example: */
| /* CALL PGM(MIGRATECR) PARM(X'1C23456789ABCDEF') */
| /* */
| /* Note: This program assumes the device to be used is */
| /* already identified either by defaulting to the CRP01 */
| /* device or by being explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(MIGRATECR) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(MIGRATECR) MODULE(MIGRATECR) */

```



```

| /*      BNDSRVPGM(QCCA/CSNBKIM QCCA/CSNBKPI QCCA/CSNBKRC      */
| /*      QCCA/CSNBDEC QCCA/CSNBKRW)                          */
| /*      */
| /* Note: Authority to the CSNBKIM, CSNBKPI, CSNBKRC, and CSNBKRW */
| /*      service programs in library QCCA is assumed.          */
| /*      */
| /*      */
| /* The Common Cryptographic Architecture (CCA) verbs used are: */
| /*      */
| /*      Key_Import (CSNBKIM)                                */
| /*      Key_Part_Import (CSNBKPI)                          */
| /*      Key_Record_Create (CSNBKRC)                        */
| /*      Key_Record_Write (CSNBKRW)                         */
| /*      */
| /*      */
| /*******/
|
| /*******/
| /* Retrieve various structures/utilities that are used in program. */
| /*******/
|
| #include <stdio.h>          /* Standard I/O header.      */
| #include <stdlib.h>        /* General utilities.       */
| #include <stddef.h>        /* Standard definitions.    */
| #include <string.h>        /* String handling utilities.*/
| #include "miptrnam.h"      /* MI templates for pointer */
|                             /* resolution instructions.  */
| #include "csucincl.h"     /* Header file for security API */
|
| /*******/
| /* Declare function prototype to build tokens to import keys */
| /*******/
| int buildImporter(char * token,
|                  char * clearkey,
|                  char * preXORcv,
|                  char * variant);
|
| /*******/
| /* Declare function prototype to import a non-CCA key and put it */
| /* into key store. */
| /*******/
| int importNonCCA(char * label,
|                 char * left_importer,
|                 char * right_importer,
|                 char * cv,
|                 char * encrypted_key);
|
| /*******/
| /* Declares for working with files */
| /*******/
| #include <xxfdbk.h>        /* Feedback area structures. */
| #include <recio.h>        /* Record I/O routines       */
| _RFILE                   *dbfptr;    /* Pointer to database file.  */
| _RIOFB_T                 *db_fdbk;   /* I/O Feedback - data base file */
| _XXOPFB_T                *db_opfb;
|
| /*******/
| /* Define the record for cross domain key file QACRKTBL */
| /*******/
| struct
| {
|     char label[10];
|     char key_type;
|     char key_value[8];
| } key_rec;

```

```

| /*****
| /* Define the structure for key tokens */
| /*****/
| typedef struct
| {
|     char    tokenType;
|     char    reserved1;
|     char    MasterKeyVerifPattern[2];
|     char    version;
|     char    reserved2;
|     char    flagByte1;
|     char    flagByte2;
|     char    reserved3[8];
|     char    leftHalfKey[8];
|     char    rightHalfKey[8];
|     char    controlVectorBase[8];
|     char    rightControlVector[8];
|     char    reserved4[12];
|     char    tvv[4];
| } key_token_T;
|
| /*****/
| /* Declare control vectors used for building keys */
| /*****/
| char        pingenc_cv[16] = { 0x00, 0x22, 0x7E, 0x00,
|                                0x03, 0x41, 0x00, 0x00,
|                                0x00, 0x22, 0x7E, 0x00,
|                                0x03, 0x21, 0x00, 0x00};
|
| char        ipinenc_cv[16] = { 0x00, 0x21, 0x5F, 0x00,
|                                0x03, 0x41, 0x00, 0x00,
|                                0x00, 0x21, 0x5F, 0x00,
|                                0x03, 0x21, 0x00, 0x00};
|
| char        opinenc_cv[16] = { 0x00, 0x24, 0x77, 0x00,
|                                0x03, 0x41, 0x00, 0x00,
|                                0x00, 0x24, 0x77, 0x00,
|                                0x03, 0x21, 0x00, 0x00};
|
| char        importer_cv[16] = { 0x00, 0x42, 0x7D, 0x00,
|                                0x03, 0x41, 0x00, 0x00,
|                                0x00, 0x42, 0x7D, 0x00,
|                                0x03, 0x21, 0x00, 0x00};
|
| char        exporter_cv[16] = { 0x00, 0x41, 0x7D, 0x00,
|                                0x03, 0x41, 0x00, 0x00,
|                                0x00, 0x41, 0x7D, 0x00,
|                                0x03, 0x21, 0x00, 0x00};
|
| char        importer_cv_part[16] = { 0x00, 0x42, 0x7D, 0x00,
|                                       0x03, 0x48, 0x00, 0x00,
|                                       0x00, 0x42, 0x7D, 0x00,
|                                       0x03, 0x28, 0x00, 0x00};
|
| char        exporter_cv_part[16] = { 0x00, 0x41, 0x7D, 0x00,
|                                       0x03, 0x48, 0x00, 0x00,
|                                       0x00, 0x41, 0x7D, 0x00,
|                                       0x03, 0x28, 0x00, 0x00};
|
| /*****/
| /* Start of mainline code. */
| /*****/
| int main(int argc, char *argv[])
| {
|     long        i,j,k;                /* Indexes for loops */
|     char        key_label[64];        /* label of new key */
|     char        key_label1[64];       /* label of new key */

```

```

| /*****
| /* Declare importer keys - two keys are needed for each type */
| /*****/
| char      EXPORTER_importerL[64];
| char      EXPORTER_importerR[64];
| char      OPINENC_importerL[64];
| char      OPINENC_importerR[64];
| char      IMPORTER_importerL[64];
| char      IMPORTER_importerR[64];
| char      PINGEN_importerL[64];
| char      PINGEN_importerR[64];
| char      IPINENC_importerL[64];
| char      IPINENC_importerR[64];
|
| /*****/
| /* Declare variables to hold bit strings to generate master key */
| /* variants. */
| /*****/
| char      variant1[16];
| char      variant2[16];
| char      variant3[16];
|
| /*****/
| /* Build the key tokens for each of the importer keys using */
| /* Key-Token_Build. Each key is built by using a variant, a control */
| /* vector, and the clear key. Master key variant 1 is the result of */
| /* an exclusive OR of the master key with hex '8888888888888888', */
| /* Master key variant 2 is the result of an exclusive OR of the */
| /* master key with hex '2222222222222222', and Master key variant 3 */
| /* is the result of an exclusive OR of the master key with hex */
| /* '4444444444444444'. During the import operation, the control */
| /* vector is exclusive OR'ed with the importer key. The effect of */
| /* the control vector is overcome by including the control vector as */
| /* key part. Then when the import operation is done, the exclusive */
| /* OR operation will result in the original key. For double keys, */
| /* the left and right half of the control vector is not the same and */
| /* therefore, XORing with the control vector will not result in the */
| /* original key - only one half of it will be valid. So two keys are */
| /* needed - one for each half. */
| /*****/
| memset(variant1, 0x88, 16);
| memset(variant2, 0x22, 16);
| memset(variant3, 0x44, 16);
|
| if (buildImporter(EXPORTER_importerL, argv[1],
|                  exporter_cv, variant1)    ||
|
|     buildImporter(EXPORTER_importerR, argv[1],
|                  &exporter_cv[8], variant1) ||
|
|     buildImporter(IMPORTER_importerL, argv[1],
|                  importer_cv, variant2)    ||
|
|     buildImporter(IMPORTER_importerR, argv[1],
|                  &importer_cv[8], variant2) ||
|
|     buildImporter(PINGEN_importerL, argv[1],
|                  pingenc_cv, variant3)    ||
|
|     buildImporter(PINGEN_importerR, argv[1],
|                  &pingenc_cv[8], variant3) ||
|
|     buildImporter(IPINENC_importerL, argv[1],
|                  ipinenc_cv, variant3)    ||

```

```

|         buildImporter(IPINENC_importerR, argv[1],
|             &ipinenc_cv[8], variant3)           ||
|
|         buildImporter(OPINENC_importerL, argv[1],
|             opinenc_cv, variant1)               ||
|
|         buildImporter(OPINENC_importerR, argv[1],
|             &opinenc_cv[8], variant1)
|
|     {
|         printf("An error occured creating the importer keys\n");
|         return;
|     }
|
| /*****
| /* Open database file.
| /*****
|
|             /* Open the input file.
|             /* If the file pointer,
|             /* dbfptr is not NULL,
|             /* then the file was
|             /* successfully opened.
| if (( dbfptr = _Ropen("QUSRSYS/QACRKTBL", "rr riofb=n"))
|     != NULL)
| {
|     db_opfb = _Ropnfbk( dbfptr );
|             /* Get pointer to the
|             /* File open feedback
|             /* area.
|
|     j = db_opfb->num_records;
|             /* Save number of records*/
|
| /*****
| /* Read keys and migrate to key storage.
| /*****
| for (i=1; i<=j; i++)
|     {
|         db_fdbk = _Rreadn(dbfptr, &key_rec,
|             sizeof(key_rec), __DFT);
|
| /*****
| /* Generate a key label for the imported keys.
| /* The key label will be similar to the label that was used for
| /* the QACRKTBL file. If the label is longer than 8 characters,
| /* then a period '.' will be inserted at position 8 to make it
| /* conform to label naming conventions for CCA. Also one
| /* one character will be added to the end to indicate what type
| /* of key. 5722-CR1 does not require unique key names across all
| /* key types. CCA requires unique labels for all keys.
| /*****
|     memset((char *)key_label, ' ',64); /* Initialize key label
|                                     /* to all blanks.
|
|     /* Copy first bytes of label
|     memcpy((char *)key_label,(char *)key_rec.label,8);
|
|     /* If label is longer than 8 characters, add a second element*/
|     if (key_rec.label[8] != ' ')
|     {
|         key_label[8] = '.';
|         key_label[9] = key_rec.label[8];
|         key_label[10] = key_rec.label[9];
|     }
|
|     /* *SND keys and *PIN keys need to be imported twice so

```

```

|      /* make a second label */
|      if (key_rec.key_type != 'R')
|          memcpy((char *)key_label1,(char *)key_label,64);
|
|      /* Add keytype to label name. Search until a space is found */
|      /* and if less than 8, add the 1 character keytype. If it */
|      /* is greater than 8, add a second element with the keytype */
|      /* 'R' is *RCV key, 'S' is *SND key, 'P' is *PIN key, */
|      /* 'I' is an IPINENC key and 'O' is OPINENC key */
|      for (k=1; k<=11; k++)
|      {
|          if (key_label[k] == ' ')
|          {
|              if (k != 8)
|              {
|                  key_label[k] = key_rec.key_type;
|
|                  /* If this is a *SND or *PIN key, update the keytype */
|                  /* in the second label as well */
|                  if (key_rec.key_type != 'R')
|                  {
|                      memcpy((char *)key_label1,(char *)key_label,64);
|                      if (key_rec.key_type == 'S')
|                          key_label1[k] = 'O';
|                      else
|                          key_label1[k] = 'I';
|                  }
|              }
|          }
|          else
|          {
|              key_label[8] = '.';
|              key_label[9] = key_rec.key_type;
|
|              /* If this is a *SND or *PIN key, update the keytype */
|              /* in the second label as well */
|              if (key_rec.key_type != 'R')
|              {
|                  memcpy((char *)key_label1,(char *)key_label,64);
|                  if (key_rec.key_type == 'S')
|                      key_label1[9] = 'O';
|                  else
|                      key_label1[9] = 'I';
|              }
|          }
|          k = 11;
|      }
|  }
|
|      /*****
|      /* Check for the type of key that was in the QACRKTBL file */
|      /* - S for SENDER key will become two keys - EXPORTER and OPINENC*/
|      /* - R for RECEIVER key will become IMPORTER key */
|      /* - P for PIN will become two keys - PINGEN and IPINENC */
|      /* Set the key id to the key token that contains the key under */
|      /* which the key in QACRKTBL is enciphered. */
|      /* Set the key_type SAPI parameter for the Secure_Key_Import verb*/
|      /*****
|      if (key_rec.key_type == 'S')
|      {
|          /* Import the exporter key */
|          if(importNonCCA(key_label,
|                          EXPORTER_importerL,
|                          EXPORTER_importerR,
|                          exporter_cv,
|                          key_rec.key_value))
|          {

```

```

        printf("An error ocured importing an exporter key¥n");
        break;
    }

    /* Import the OPINENC key          */
    if (importNonCCA(key_label1,
                    OPINENC_importerL,
                    OPINENC_importerR,
                    opinenc_cv,
                    key_rec.key_value))
    {
        printf("An error ocured importing an opinenc key¥n");
        break;
    }
}
else
if (key_rec.key_type == 'R')
{
    /* Import the importer key          */
    if (importNonCCA(key_label,
                    IMPORTER_importerL,
                    IMPORTER_importerR,
                    importer_cv,
                    key_rec.key_value))
    {
        printf("An error ocured importing an importer key¥n");
        break;
    }
}
else
{
    /* Import the PINGEN key          */
    if(importNonCCA(key_label,
                    PINGEN_importerL,
                    PINGEN_importerR,
                    pingenc_cv,
                    key_rec.key_value))
    {
        printf("An error ocured importing a PINGEN key¥n");
        break;
    }

    /* Import the IPINENC key          */
    if(importNonCCA(key_label1,
                    IPINENC_importerL,
                    IPINENC_importerR,
                    ipinenc_cv,
                    key_rec.key_value))
    {
        printf("An error ocured importing an ipinenc key¥n");
        break;
    }
}

}          /* End loop repeating for each record */

/*****
/* Close database file.          */
*****/
if (dbfptr != NULL)          /* Close the file.          */
    _Rclose(dbfptr);

}          /* End if file open leg          */
else
{
    printf("An error ocured opening the QACRKTBL file.¥n");
}

```

```

|     }
| }                                     /* End of main()      */
|
|
| /*****
| /* buildImporter creates an importer token from a clearkey exclusive*/
| /* OR'ed with a variant and a control vector. The control vector */
| /* is XOR'ed in order to import non-CCA keys. The variant is XOR'ed*/
| /* in order to import from implementations that use different */
| /* master key variants to protect keys as does 5722-CR1.      */
| /*****
| int buildImporter(char * token,
|                   char * clearkey,
|                   char * preXORcv,
|                   char * variant)
| {
| /*****
| /* Declare variables used by the SAPI's */
| /*****
| char      rule_array[16];
| long      rule_array_count;
| long      return_code;
| long      reason_code;
| long      exit_data_length;
| char      exit_data[4];
| char      keyvalue[16];
| char      keytype[8];
| char      ctl_vector[16];
| key_token_T *token_ptr;
|
| /*****
| /* Build an IMPORTER token */
| /*****
| memset(token, 0, 64);          /* Initialize token to all 0's */
| token_ptr = (key_token_T *)token;
| token_ptr->tokenType = 0x01;    /* 01 is internal token */
| token_ptr->version = 0x03;     /* Version 3 token */
| token_ptr->flagByte1 = 0x40;   /* High order bit is 0 so key */
|                               /* is not present. The 40 */
|                               /* bit means that CV is present*/
|
|                               /* Copy control vector into */
|                               /* the token. */
| memcpy(token_ptr->controlVectorBase, importer_cv_part, 16);
|                               /* Copy TVV into token. This */
|                               /* was calculated manually by */
|                               /* setting all the fields and */
|                               /* then adding each 4 bytes of */
|                               /* the token (excluding the */
|                               /* TVV) together. */
| memcpy(token_ptr->tvv, "\x0A\xF5\x3A\x00", 4);
|
| /*****
| /* Import the control vector as a key part using Key_Part_Import */
| /*****
| exit_data_length = 0;
| rule_array_count = 1;
| memcpy(ctl_vector, preXORcv, 8);
| memcpy(&ctl_vector[8], preXORcv, 8); /* Need to copy the
|                                     control vector into the
|                                     second 8 bytes as well*/
|
| memcpy(rule_array, "FIRST ", 8);
| CSNBKPI( &return_code, &reason_code, &exit_data_length,
|         (char *) exit_data,
|         (long *) &rule_array_count,

```

```

        (char *) rule_array,
        (char *) ctl_vector,
        (char *) token);

    if (return_code > 4)
    {
        printf("Key_Part_Import failed with return/reason codes ¥
                %d/%d ¥n",return_code, reason_code);
        return 1;
    }

/*****
/* Import the variant as a key part using Key_Part_Import      */
/*****
    memcpy(rule_array, "MIDDLE ", 8);
    CSNBKPI( &return_code, &reason_code, &exit_data_length,
            (char *) exit_data,
            (long *) &rule_array_count,
            (char *) rule_array,
            (char *) variant,
            (char *) token);

    if (return_code > 4)
    {
        printf("Key_Part_Import failed with return/reason codes ¥
                %d/%d ¥n",return_code, reason_code);
        return 1;
    }

/*****
/* Import the clear key as a key part using Key_Part_Import  */
/*****
    memcpy(keyvalue, clearkey, 8);
    memcpy(&keyvalue[8], clearkey, 8); /* Make key double length*/
    memcpy(rule_array, "LAST ", 8);
    CSNBKPI( &return_code, &reason_code, &exit_data_length,
            (char *) exit_data,
            (long *) &rule_array_count,
            (char *) rule_array,
            (char *) keyvalue,
            (char *) token);

    if (return_code > 4)
    {
        printf("Key_Part_Import failed with return/reason codes ¥
                %d/%d ¥n",return_code, reason_code);
        return 1;
    }

    return 0;
}

/*****
/* importNonCCA imports a double length key into CCA from the  */
/* non-CCA implementation                                       */
/*****
int importNonCCA(char * label,
                char * left_importer,
                char * right_importer,
                char * cv,
                char * encrypted_key)
{
/*****
/* Declare vaiables used by the SAPIs      */
/*****
long                return_code, reason_code;

```



```

| char          exit_data[4];
| long         exit_data_length;
| long         rule_array_count;
| char         rule_array[24];
| char         keytoken[64];
| char         externalkey[64];
| char         keyvalue[16];
| char         keytype[8];
| char         *importer;
| char         mkvp[2];
| key_token_T  *token_ptr;
| int          tvv, tvv_part;
| char         *tvv_pos;
|
| /*****
| /* Build an external key token to IMPORT from */
| *****/
| memset((void *)externalkey, '¥00', 64);
| token_ptr = (key_token_T *)externalkey;
| token_ptr->tokenType = 0x02;          /* 02 is external token */
| token_ptr->version = 0x00;           /* Version 0 token */
| token_ptr->flagByte1 = 0xC0;         /* High order bit is 1 so */
|                                     /* key is present. The */
|                                     /* 40 bit means that CV */
|                                     /* is present */
|
| memcpy(token_ptr->controlVectorBase, cv, 16); /* Copy control
|                                             vector into token */
| memcpy(token_ptr->leftHalfKey, encrypted_key, 8); /* Copy key
|                                             into left half */
| memcpy(token_ptr->rightHalfKey, encrypted_key, 8); /* Copy key
|                                             into right half */
|
| /*****
| /* Calculate the TVV by adding every 4 bytes */
| *****/
| tvv_pos = externalkey;
| tvv = 0;
| while (tvv_pos < (externalkey + 60))
| {
|     memcpy((void*)&tvv_part, tvv_pos, 4);
|     tvv += tvv_part;
|     tvv_pos += 4;
| }
| memcpy(token_ptr->tvv, (void*)&tvv, 4);
|
| /*****
| /* Import the left half of the key using Key_Import and */
| /* the importer built with left half of the control vector */
| *****/
| exit_data_length = 0;
| memcpy(keytype, "TOKEN ", 8);
| memset((void *)keytoken, '¥00', 64);
| CSNBKIM( &return_code, &reason_code, &exit_data_length,
|         (char *) exit_data,
|         (char *) keytype,
|         (char *) externalkey,
|         (char *) left_importer,
|         (char *) keytoken);
|
| if (return_code > 4)
| {
|     printf("Key_Import failed with return/reason codes ¥
|           %d/%d ¥n", return_code, reason_code);
|     return 1;
| }

```

```

| /*****
| /* Save left half of key out of key token */
| /*****/
|     memcpy(keyvalue, &keytoken[16], 8);
|
| /*****
| /* Import the right half of the key using Key_Import and */
| /* the importer built with right half of the control vector*/
| /*****/
|     memcpy(keytype, "TOKEN ", 8);
|     memset((void *)keytoken, '\0', 64);
|     CSNBKIM( &return_code, &reason_code, &exit_data_length,
|             (char *) exit_data,
|             (char *) keytype,
|             (char *) externalkey,
|             (char *) right_importer,
|             (char *) keytoken);
|
|     if (return_code > 4)
|     {
|         printf("Key_Import failed with return/reason codes ¥
|             %d/%d ¥n", return_code, reason_code);
|         return 1;
|     }
|
| /*****
| /* Save right half of key out of key token */
| /*****/
|     memcpy(&keyvalue[8], &keytoken[24], 8);
|
| /*****
| /* Get master key verification pattern from the last key token built */
| /*****/
|     mkvp[0] = keytoken[2];
|     mkvp[1] = keytoken[3];
|
| /*****
| /* Build an internal key token using both key halves just */
| /* imported and using the master key verification pattern */
| /*****/
|     memset((void *)keytoken, '\0', 64);
|     exit_data_length = 0;
|     token_ptr = (key_token_T *)keytoken;
|     token_ptr->tokenType = 0x01;          /* 01 is internal token */
|     token_ptr->version = 0x03;           /* Version 3 token */
|     token_ptr->flagByte1 = 0xC0;        /* High order bit is 1 so */
|                                         /* key is present. The */
|                                         /* 40 bit means that CV is */
|                                         /* present */
|
|                                         /* Set the first byte of */
|                                         /* Master key verification */
|                                         /* pattern. */
|     token_ptr->MasterKeyVerifPattern[0] = mkvp[0];
|                                         /* Set the second byte of */
|                                         /* Master key verification */
|                                         /* pattern. */
|     token_ptr->MasterKeyVerifPattern[1] = mkvp[1];
|
|                                         /* Copy control vector into*/
|                                         /* token */
|     memcpy(token_ptr->controlVectorBase, cv, 16);
|     memcpy(token_ptr->leftHalfKey, keyvalue, 16); /*Copy key to token */

```

```

|
| /*****
| /* Calculate the TVV by adding every 4 bytes */
| /*****
| tvv_pos = externalkey;
| tvv = 0;
| while (tvv_pos < (externalkey + 60))
| {
|     memcpy((void*)&tvv_part,tvv_pos,4);
|     tvv += tvv_part;
|     tvv_pos += 4;
| }
| memcpy(token_ptr->tvv, (void*)&tvv, 4);
|
| /*****
| /* Create a Key Record in Key Store */
| /*****
| exit_data_length = 0;
| CSNBKRC((long *) &return_code,
|         (long *) &reason_code,
|         (long *) &exit_data_length,
|         (char *) exit_data,
|         (char *) label);
|
| if (return_code > 4)
| {
|     printf("Key_Record_Create failed with return/reason codes ¥
|           %d/%d ¥n",return_code, reason_code);
|     return 1;
| }
|
| /*****
| /* Write the record out to Key Store */
| /*****
| CSNBKRW((long *) &return_code,
|         (long *) &reason_code,
|         (long *) &exit_data_length,
|         (char *) exit_data,
|         (char *) keytoken,
|         (char *) label);
|
| if (return_code > 4)
| {
|     printf("Key_Record_Write failed with return/reason codes ¥
|           %d/%d ¥n",return_code, reason_code);
|     return 1;
| }
|
| return 0;
| }

```

インポーター鍵暗号鍵の使用:

すべてのクロスドメイン鍵をインポートするために必要なすべてのインポーター鍵暗号鍵の要約については、ここに記載された情報を参照してください。ここでは、そのインポーター鍵暗号鍵の作成方法も示しています。

あらゆるタイプのクロスドメイン鍵をインポートするには、次の **IMPORTER** 鍵暗号化鍵が必要です。

1. エクスポーター鍵の左半分をインポートするための **KEK**。

この鍵は、クリア・ホスト・マスター鍵、エクスポーター鍵暗号化鍵制御ベクトルの左半分、および 16 進 88 の 16 バイトを使用して作成します。

2. エクスポーター鍵の右半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、エクスポーター鍵暗号化鍵制御ベクトルの右半分、および 16 進 88 の 16 バイトを使用して作成します。
3. インポーター鍵の左半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、インポーター鍵暗号化鍵制御ベクトルの左半分、および 16 進 22 の 16 バイトを使用して作成します。
4. インポーター鍵の右半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、インポーター鍵暗号化鍵制御ベクトルの右半分、および 16 進 22 の 16 バイトを使用して作成します。
5. OPINENC 鍵の左半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、OPINENC 鍵制御ベクトルの左半分、および 16 進 88 の 16 バイトを使用して作成します。
6. OPINENC 鍵の右半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、OPINENC 鍵制御ベクトルの右半分、および 16 進 88 の 16 バイトを使用して作成します。
7. IPINENC 鍵の左半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、IPINENC 鍵制御ベクトルの左半分、および 16 進 44 の 16 バイトを使用して作成します。
8. IPINENC 鍵の右半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、IPINENC 鍵制御ベクトルの右半分、および 16 進 44 の 16 バイトを使用して作成します。
9. PINGEN 鍵の左半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、PINGEN 鍵制御ベクトルの右半分、および 16 進 44 の 16 バイトを使用して作成します。
10. PINGEN 鍵の右半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、PINGEN 鍵制御ベクトルの右半分、および 16 進 44 の 16 バイトを使用して作成します。
11. PINVER 鍵の左半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、PINVER 鍵制御ベクトルの左半分、および 16 進 44 の 16 バイトを使用して作成します。
12. PINVER 鍵の右半分をインポートするための KEK。
この鍵は、クリア・ホスト・マスター鍵、PINVER 鍵制御ベクトルの左半分、および 16 進 44 の 16 バイトを使用して作成します。

暗号化コプロセッサの管理

- 暗号化コプロセッサをセットアップした後で、暗号化コプロセッサの暗号機能を利用するためプログラムの作成を開始することができます。

| このセクションでは主に、暗号化コプロセッサを i5/OS アプリケーションで使用する場合について説明
| します。

| 注: このセクションにあるページの多くには、1 つ以上のプログラム例が示してあります。必要に応じてこ
| れらのプログラムを変更してください。パラメーターを 1 つか 2 つ変更するのみでよい場合もあれ
| ば、広範囲にわたって変更する必要がある場合もあります。セキュリティ上の理由から、IBM で
| は、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使
| 用することをお勧めします。

| 暗号化コプロセッサのログオンまたはログオフ

| 暗号化コプロセッサにログオンまたはログオフするには、役割制限のある API を使用します。

| ログオン

| デフォルトの役割で使用可能になっていないアクセス制御ポイントを使用する API を使用する場合にはのみ
| ログオンする必要があります。使用したいアクセス制御ポイントが使用可能になっている役割を使用するプ
| ロファイルでログオンします。

| 暗号化コプロセッサにログオンした後で、暗号化コプロセッサ用の暗号機能を利用するためのプログラ
| ムを実行することができます。Logon_Control (CSUALCT) API verb を使用するアプリケーションを作成し
| てログオンすることもできます。

| ログオフ

| 暗号化コプロセッサを終了したときは、暗号化コプロセッサからログオフする必要があります。
| Logon_Control (CSUALCT) API verb を使用するアプリケーションを作成してログオフすることもできま
| す。

| 注:

| 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参
| 照してください。

| 関連概念

| 149 ページの『DES キーおよび PKA キーの作成』
| DES および PKA 鍵を作成して、DES 鍵ストアに保管することができます。

| 例: 暗号化コプロセッサにログオンするための ILE C プログラム:

| 暗号化コプロセッサにログオンするには、必要に応じて以下のプログラム例を変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
| ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これ
| らのプログラム例を修正して使用することをお勧めします。

```
| /*-----*/  
| /* Log on to the card using your profile and passphrase. */  
| /* */  
| /* */  
| /* COPYRIGHT 5769-SS1, 5722-SS1 (C) IBM CORP. 1999, 2000 */  
| /* */  
| /* This material contains programming source code for your */
```

```

| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* */
| /* Note: This verb is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* none. */
| /* */
| /* Example: */
| /* CALL PGM(LOGON) */
| /* */
| /* Note: This program assumes the card with the profile is */
| /* already identified either by defaulting to the CRP01 */
| /* device or by being explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(LOGON) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(LOGON) MODULE(LOGON) BNDSRVPGM(QCCA/CSUALCT) */
| /* */
| /* Note: Authority to the CSUALCT service program in the */
| /* QCCA library is assumed. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Logon_Control (CSUALCT). */
| /* */
| /*-----*/
|
| #include "csucincl.h" /* header file for CCA Cryptographic */
| /* Service Provider */
|
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
| #define WARNING 4
|
| int main(int argc, char *argv[])
|
| {
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code = 0;
| long reason_code = 0;

```

```

| long exit_data_length = 2;
| char exit_data[4];
| char rule_array[2][8];
| long rule_array_count = 2;
|
| /*-----*/
| /* fields unique to this sample program */
| /*-----*/
|
| char profile[8];
| long auth_parm_length;
| char auth_parm[4];
| long auth_data_length;
| char auth_data[256];
|
| /* set rule array keywords */
| memcpy(rule_array,"LOGON PPHRASE ", 16);
|
| /* Check for correct number of parameters */
| if (argc < 3)
| {
|     printf("Usage: CALL LOGON ( profile 'pass phrase')\n");
|     return(ERROR);
| }
|
| /* Set profile and pad out with blanks */
| memset(profile, ' ', 8);
| if (strlen(argv[1]) > 8)
| {
|     printf("Profile is limited to 8 characters.\n");
|     return(ERROR);
| }
| memcpy(profile, argv[1], strlen(argv[1]));
|
| /* Authentication parm length must be 0 for logon */
| auth_parm_length = 0;
|
| /* Authentication data length is length of the pass-phrase */
| auth_data_length = strlen(argv[2]);
|
|
| /* invoke verb to log on to the card */
|
| CSUALCT( &return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (char *)rule_array,
|         profile,
|         &auth_parm_length,
|         auth_parm,
|         &auth_data_length,
|         argv[2]);
|
| if (return_code != OK)
| {
|     printf("Log on failed with return/reason codes %ld/%ld\n",
|           return_code, reason_code);
| }
| else
|     printf("Logon was successful\n");
| }

```

| **例: 暗号化コプロセッサにログオンするための ILE RPG プログラム:**

| 暗号化コプロセッサにログオンするには、必要に応じて以下のプログラム例を変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
| ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これ
| らのプログラム例を修正して使用することをお勧めします。

```
| D*****  
| D* LOGON  
| D*  
| D* Log on to the Cryptographic Coprocessor.  
| D*  
| D*  
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000  
| D*  
| D* This material contains programming source code for your  
| D* consideration. These example has not been thoroughly  
| D* tested under all conditions. IBM, therefore, cannot  
| D* guarantee or imply reliability, serviceability, or function  
| D* of these programs. All programs contained herein are  
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF  
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for  
| D* these programs and files.  
| D*  
| D*  
| D* Note: Input format is more fully described in Chapter 2 of  
| D* IBM CCA Basic Services Reference and Guide  
| D* (SC31-8609) publication.  
| D*  
| D* Parameters: Profile  
| D* Pass-phrase  
| D*  
| D* Example:  
| D* CALL PGM(LOGON) PARM(PROFILE PASSPRHASE)  
| D*  
| D* Use these commands to compile this program on the system:  
| D* CRTRPGMOD MODULE(LOGON) SRCFILE(SAMPLE)  
| D* CRTPGM PGM(LOGON) MODULE(LOGON)  
| D* BNDDIR(QCCA/QC6BNDDIR)  
| D*  
| D* Note: Authority to the CSUALCT service program in the  
| D* QCCA library is assumed.  
| D*  
| D* The Common Cryptographic Architecture (CCA) verbs used are  
| D* Cryptographic_Facilty_Control (CSUACFC)  
| D*  
| D* This program assumes the card with the profile is  
| D* already identified either by defaulting to the CRP01  
| D* device or by being explicitly named using the  
| D* Cryptographic_Resource_Allocate verb. Also this  
| D* device must be varied on and you must be authorized  
| D* to use this device description.  
| D*****  
| D*-----  
| D* Declare variables for CCA SAPI calls  
| D*-----  
| D* ** Return code  
| DRETURNCODE S 9B 0  
| D* ** Reason code  
| DREASONCODE S 9B 0
```



```

|      D*          ** Exit data length
|      DEXITDATALEN S          9B 0
|      D*          ** Exit data
|      DEXITDATA    S          4
|      D*          ** Rule array count
|      DRULEARRAYCNT S         9B 0
|      D*          ** Rule array
|      DRULEARRAY   S          16
|      D*          ** Userid parm
|      DUSERID      S          8
|      D*          ** Authentication parameter length
|      DAUTHPARMLEN S         9B 0 INZ(0)
|      D*          ** Authentication parameter
|      DAUTHPARM    S          10
|      D*          ** Authentication data length
|      DAUTHDATALEN S         9B 0 INZ(0)
|      D*          ** Authentication data
|      DAUTHDATA    S          50
|      D*
|      D*****
|      D* Prototype for Logon Control (CSUALCT)
|      D*****
|      DCSUALCT      PR
|      DRETCODE      9B 0
|      DRNSCODE      9B 0
|      DEXTDTALEN    9B 0
|      DEXTDTA       4
|      DRARRAYCT     9B 0
|      DRARRAY       16
|      DUSR          8
|      DATHPRMLEN    9B 0
|      DATHPRM       10
|      DATHDTALEN    9B 0
|      DATHDTA       50
|      D*
|      D*****
|      D* Declares for sending messages to job log
|      D*****
|      D*-----
|      D*          ** Declares for sending messages to the
|      D*          ** job log using the QMHSDPM API
|      D*-----
|      DMSG          S          75   DIM(2) CTDATA PERRCD(1)
|      DMSGLENGTH    S          9B 0 INZ(75)
|      D             DS
|      DMSGTEXT      1          75
|      DFAILRETC     41         44
|      DFAILRSNC     46         49
|      DMESSAGEID    S          7   INZ(' ')
|      DMESSAGEFILE  S          21  INZ(' ')
|      DMSGKEY       S          4   INZ(' ')
|      DMSGTYPE      S          10  INZ('*INFO ')
|      DSTACKENTRY   S          10  INZ('* ')
|      DSTACKCOUNTER S         9B 0 INZ(2)
|      DERRCODE      DS
|      DBYTESIN      1          4B 0 INZ(0)
|      DBYTESOUT     5          8B 0 INZ(0)
|      D*
|      C*****
|      C* START OF PROGRAM *
|      C* *
|      C*-----*
|      C   *ENTRY      PLIST
|      C               PARM          USERID
|      C               PARM          AUTHDATA
|      C*-----*
|      C* Set the keywords in the rule array *

```

```

C*-----*
C          MOVE      'LOGON  '    RULEARRAY
C          MOVE      'PPHRASE '    RULEARRAY
C          Z-ADD     2             RULEARRAYCNT
C*-----*
C* Get the length of the passphrase *
C*-----*
C          EVAL      AUTHDATALEN = %LEN(%TRIM(AUTHDATA))
C*
C*****
C* Call Logon Control SAPI
C*****
C          CALLP     CSUALCT      (RETURNCODE:
C                                REASONCODE:
C                                EXITDATALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                USERID:
C                                AUTHPARMLEN:
C                                AUTHPARM:
C                                AUTHDATALEN:
C                                AUTHDATA)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      0
C*          *-----*
C*          * Send error message *
C*          *-----*
C          MOVE      MSG(1)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC
C          MOVE      REASONCODE  FAILRSNC
C          EXSR      SNDMSG
C*
C          ELSE
C*          *-----*
C*          * Send success message *
C*          *-----*
C          MOVE      MSG(2)      MSGTEXT
C          EXSR      SNDMSG
C*
C          ENDIF
C*
C          SETON
C
C*-----*
C* Subroutine to send a message
C*****
C          SNDMSG     BEGSR
C          CALL      'QMHSNDPM'
C          PARM      MESSAGEID
C          PARM      MESSAGEFILE
C          PARM      MSGTEXT
C          PARM      MSGLENGTH
C          PARM      MSGTYPE
C          PARM      STACKENTRY
C          PARM      STACKCOUNTER
C          PARM      MSGKEY
C          PARM      ERRCODE
C          ENDSR
C*

```

LR

```

**
CSUALCT failed with return/reason codes 9999/9999'
The request completed successfully

```

例: 暗号化コプロセッサからログオフするための ILE C プログラム:

暗号化コプロセッサからログオフするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
/*-----*/
/* Log off the Cryptographic CoProcessor          */
/*                                                */
/*                                                */
/* COPYRIGHT 5769-SS1, 5722-SS1 (C) IBM CORP. 1999, 2000 */
/*                                                */
/* This material contains programming source code for your */
/* consideration. These examples have not been thoroughly */
/* tested under all conditions. IBM, therefore, cannot */
/* guarantee or imply reliability, serviceability, or function */
/* of these program. All programs contained herein are */
/* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
/* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
/* these programs and files. */
/*                                                */
/* Note: This verb is more fully described in Chapter 2 of */
/* IBM CCA Basic Services Reference and Guide */
/* (SC31-8609) publication. */
/* Parameters: */
/* none. */
/* Example: */
/* CALL PGM(LOGOFF) */
/* Note: This program assumes the card with the profile is */
/* already identified either by defaulting to the CRP01 */
/* device or by being explicitly named using the */
/* Cryptographic_Resource_Allocate verb. Also this */
/* device must be varied on and you must be authorized */
/* to use this device description. */
/* Use these commands to compile this program on the system: */
/* ADDLIB LIB(QCCA) */
/* CRTCMOD MODULE(LOGOFF) SRCFILE(SAMPLE) */
/* CRTPGM PGM(LOGOFF) MODULE(LOGOFF) BNDSRVPGM(QCCA/CSUALCT) */
/* Note: Authority to the CSUALCT service program in the */
/* QCCA library is assumed. */
/* The Common Cryptographic Architecture (CCA) verb used is */
/* Logon_Control (CSUALCT). */
/*-----*/

#include "csucincl.h" /* header file for CCA Cryptographic */
/* Service Provider */
#include <stdio.h>
#include <string.h>
```

```

| #include <stdlib.h>
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR    -1
| #define OK       0
|
| int main(int argc, char *argv[])
|
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[2][8];
|     long rule_array_count = 1;
|
|     /*-----*/
|     /* fields unique to this sample program */
|     /*-----*/
|     char profile[8];
|     long auth_parm_length;
|     char * auth_parm = " ";
|     long auth_data_length = 256;
|     char auth_data[300];
|
|     /* set rule array keywords to log off */
|     memcpy(rule_array,"LOGOFF ",8);
|
|     rule_array_count = 1;
|
|     /* Both Authentication parm and data lengths must be 0 */
|     auth_parm_length = 0;
|     auth_data_length = 0;
|
|     /* Invoke verb to log off the Cryptographic CoProcessor */
|     CSUALCT( &return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             &rule_array_count,
|             (char *)rule_array,
|             profile,
|             &auth_parm_length,
|             auth_parm,
|             &auth_data_length,
|             auth_data);
|
|     if (return_code != OK)
|     {
|         printf("Log off failed with return/reason codes %ld/%ld\n",
|               return_code, reason_code);
|         return(ERROR);
|     }
|     else
|     {
|         printf("Log off successful\n");
|         return(OK);
|     }
| }

```

| **例: 暗号化コプロセッサからログオフするための ILE RPG プログラム:**

| 暗号化コプロセッサからログオフするには、必要に応じて以下のプログラム例を変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
| ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これ
| らのプログラム例を修正して使用することをお勧めします。

```
| D*****  
| D* LOGOFF  
| D*  
| D* Log off from the Cryptographic Coprocessor.  
| D*  
| D*  
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000  
| D*  
| D* This material contains programming source code for your  
| D* consideration. These example has not been thoroughly  
| D* tested under all conditions. IBM, therefore, cannot  
| D* guarantee or imply reliability, serviceability, or function  
| D* of these programs. All programs contained herein are  
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF  
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for  
| D* these programs and files.  
| D*  
| D*  
| D* Note: Input format is more fully described in Chapter 2 of  
| D* IBM CCA Basic Services Reference and Guide  
| D* (SC31-8609) publication.  
| D*  
| D* Parameters: None  
| D*  
| D* Example:  
| D* CALL PGM(LOGOFF)  
| D*  
| D* Use these commands to compile this program on the system:  
| D* CRTRPGMOD MODULE(LOGOFF) SRCFILE(SAMPLE)  
| D* CRTPGM PGM(LOGOFF) MODULE(LOGOFF)  
| D* BNDDIR(QCCA/QC6BNDDIR)  
| D*  
| D* Note: Authority to the CSUALCT service program in the  
| D* QCCA library is assumed.  
| D*  
| D* The Common Cryptographic Architecture (CCA) verbs used are  
| D* Cryptographic_Facilty_Control (CSUACFC)  
| D*  
| D* This program assumes the card with the profile is  
| D* already identified either by defaulting to the CRP01  
| D* device or by being explicitly named using the  
| D* Cryptographic_Resource_Allocate verb. Also this  
| D* device must be varied on and you must be authorized  
| D* to use this device description.  
| D*****  
| D*-----  
| D* Declare variables for CCA SAPI calls  
| D*-----  
| D* ** Return code  
| DRETURNCODE S 9B 0  
| D* ** Reason code  
| DREASONCODE S 9B 0  
| D* ** Exit data length
```

```

|      DEXITDATALEN      S          9B 0
|      D*                ** Exit data
|      DEXITDATA        S          4
|      D*                ** Rule array count
|      DRULEARRAYCNT    S          9B 0
|      D*                ** Rule array
|      DRULEARRAY       S          16
|      D*                ** Userid parm
|      DUSERID          S          8
|      D*                ** Authentication parameter length
|      DAUTHPARMLEN    S          9B 0 INZ(0)
|      D*                ** Authentication parameter
|      DAUTHPARM        S          8
|      D*                ** Authentication data length
|      DAUTHDATALEN    S          9B 0 INZ(0)
|      D*                ** Authentication data
|      DAUTHDATA        S          8
|      D*
|      D*****
|      D* Prototype for Logon Control (CSUALCT)
|      D*****
|      DCSUALCT         PR
|      DRETCODE         S          9B 0
|      DRSNCODE         S          9B 0
|      DEXTDTALEN       S          9B 0
|      DEXTDTA          S          4
|      DRARRAYCT        S          9B 0
|      DRARRAY          S          16
|      DUSR             S          8
|      DATHPRMLEN       S          9B 0
|      DATHPRM          S          8
|      DATHDTALEN       S          9B 0
|      DATHDTA          S          8
|      D*-----
|      D*                ** Declares for sending messages to the
|      D*                ** job log using the QMHSNDPM API
|      D*-----
|      DMSG              S          75   DIM(2) CTDATA PERRCD(1)
|      DMSGLENGTH        S          9B 0 INZ(75)
|      D                 DS
|      DMSGTEXT          S          1    75
|      DFAILRETC         S          41   44
|      DFAILRSNC         S          46   49
|      DMESSAGEID        S          7    INZ(' ')
|      DMESSAGEFILE      S          21   INZ(' ')
|      DMSGKEY           S          4    INZ(' ')
|      DMSGTYPE          S          10   INZ('*INFO ')
|      DSTACKENTRY       S          10   INZ('* ')
|      DSTACKCOUNTER     S          9B 0 INZ(2)
|      DERRCODE          DS
|      DBYTESIN          S          1    4B 0 INZ(0)
|      DBYTESOUT         S          5    8B 0 INZ(0)
|      D*
|      C*****
|      C* START OF PROGRAM
|      C*
|      C*-----
|      C* Set the keywords in the rule array
|      C*-----
|      C                 MOVEL   'LOGOFF '   RULEARRAY
|      C                 Z-ADD   1         RULEARRAYCNT
|      C*
|      C*****
|      C* Call Logon Control SAPI
|      C*****
|      C                 CALLP   CSUALCT   (RETURNCODE:
|      C                                     REASONCODE:

```

```

|         C                               EXITDATALEN:
|         C                               EXITDATA:
|         C                               RULEARRAYCNT:
|         C                               RULEARRAY:
|         C                               USERID:
|         C                               AUTHPARMLEN:
|         C                               AUTHPARAM:
|         C                               AUTHDATALEN:
|         C                               AUTHDATA)
|
|         C*-----*
|         C* Check the return code *
|         C*-----*
|         C      RETURNCODE      IFGT      0
|         C*
|         C*      *-----*
|         C*      * Send error message *
|         C*      *-----*
|         C              MOVE      MSG(1)      MSGTEXT
|         C              MOVE      RETURNCODE  FAILRETC
|         C              MOVE      REASONCODE  FAILRSNC
|         C              EXSR      SNDMSG
|         C*
|         C              ELSE
|         C*      *-----*
|         C*      * Send success message *
|         C*      *-----*
|         C              MOVE      MSG(2)      MSGTEXT
|         C              EXSR      SNDMSG
|         C*
|         C              ENDIF
|         C*
|         C              SETON                               LR
|         C*
|         C*****
|         C* Subroutine to send a message
|         C*****
|         C      SNDMSG      BEGSR
|         C              CALL      'QMHSNDPM'
|         C              PARM      MESSAGEID
|         C              PARM      MESSAGEFILE
|         C              PARM      MSGTEXT
|         C              PARM      MSGLENGTH
|         C              PARM      MSGTYPE
|         C              PARM      STACKENTRY
|         C              PARM      STACKCOUNTER
|         C              PARM      MSGKEY
|         C              PARM      ERRCODE
|         C              ENDSR
|         C*
|
|         **
|         CSUALCT failed with return/reason codes 9999/9999'
|         The request completed successfully

```

状況の照会または情報の要求

- | 使用可能になっているアルゴリズム、サポートされている鍵の長さ、マスター鍵の状況、複製の状況、クロックの設定などの特性を判別するために、暗号化コプロセッサに照会することができます。
- | 最も簡単かつ最も迅速に暗号化コプロセッサに照会するには、暗号化コプロセッサ構成のための Web ベースのユーティリティを使用します。「構成の表示 (Display configuration)」をクリックして装置を選択し、次に表示する項目を選択します。
- | 独自のアプリケーションを作成して、コプロセッサに照会することもできます。これを行うには、Cryptographic_Facility_Query (CSUACFQ) API verb を使用します。「IBM PCI 暗号化コプロセッサ CCA 基本サービスのリファレンスおよびガイド (IBM PCI Cryptographic Coprocessor CCA Basic Services

Reference and Guide)」では、Cryptographic_Facility_Query (CSUACFQ) 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、要求できる情報のタイプ、および戻される情報の形式などについて説明しています。

例: 暗号化コプロセッサの状況の照会:

暗号化コプロセッサの状況を照会するには、必要に応じて以下のプログラム例を変更してください。このプログラムでは、STATEID および TIMEDATE キーワードを使用します。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
/*-----*/
/* Query the card for status or other information.          */
/* This sample program uses the STATEID and TIMEDATE keywords. */
/*                                                         */
/*                                                         */
/* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999                  */
/*                                                         */
/* This material contains programming source code for your */
/* consideration. These examples have not been thoroughly */
/* tested under all conditions. IBM, therefore, cannot    */
/* guarantee or imply reliability, serviceability, or function */
/* of these program. All programs contained herein are    */
/* provided to you "AS IS". THE IMPLIED WARRANTIES OF    */
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
/* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
/* these programs and files.                               */
/*                                                         */
/*                                                         */
/* Note: This verb is more fully described in Chapter 2 of */
/*       IBM CCA Basic Services Reference and Guide      */
/*       (SC31-8609) publication.                         */
/*                                                         */
/* Parameters:                                           */
/* none.                                                 */
/*                                                         */
/* Example:                                             */
/* CALL PGM(QUERY)                                       */
/*                                                         */
/* Note: This program assumes the device to use is      */
/*       already identified either by defaulting to the CRP01 */
/*       device or by being explicitly named using the    */
/*       Cryptographic_Resource_Allocate verb. Also this */
/*       device must be varied on and you must be authorized */
/*       to use this device description.                 */
/*                                                         */
/* Use these commands to compile this program on the system: */
/* ADDLIB LIB(QCCA)                                       */
/* CRTCMOD MODULE(QUERY) SRCFILE(SAMPLE)                */
/* CRTPGM PGM(QUERY) MODULE(QUERY) BNDSRVPGM(QCCA/CSUACFQ) */
/*                                                         */
/* Note: Authority to the CSUACFQ service program in the  */
/*       QCCA library is assumed.                       */
/*                                                         */
/* The Common Cryptographic Architecture (CCA) verb used is */
/* Cryptographic_Facility_Query (CSUACFQ).              */
/*                                                         */
```



```

| /*-----*/
|
| #include "csucincl.h" /* header file for CCA Cryptographic */
| /* Service Provider */
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
| #define WARNING 4
|
| #define IDSIZE 16 /* number of bytes in environment ID */
| #define TIMEDATESIZE 24 /* number of bytes in time and date */
|
| int main(int argc, char *argv[])
|
| {
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code = 0;
| long reason_code = 0;
| long exit_data_length = 2;
| char exit_data[4];
| char rule_array[2][8];
| long rule_array_count = 2;
| char rule_array2[3][8];
|
| /*-----*/
| /* fields unique to this sample program */
| /*-----*/
|
| long verb_data_length = 0; /* currently not used by this verb */
| char * verb_data = " ";
|
| /* set keywords in the rule array */
|
| memcpy(rule_array,"ADAPTER1STATEID ",16);
|
| /* get the environment ID from the card */
|
| CSUACFQ( &return_code,
| &reason_code,
| &exit_data_length,
| exit_data,
| &rule_array_count,
| (char *)rule_array,
| &verb_data_length,
| verb_data);
|
| if ( (return_code == OK) | (return_code == WARNING) )
| {
| printf("Environment ID was successfully returned.\n");
|
| printf("Return/reason codes ");
|
| printf("%1d/%1d\n\n", return_code, reason_code);
|
| printf("ID = %.16s\n", rule_array);

```

```

|     }
|
|     else
|     {
| printf("An error occurred while getting the environment ID.¥n");
| printf("Return/reason codes ");
| printf("%1d/%1d¥n¥n", return_code, reason_code);
| /* return(ERROR) */;
|     }
|
|     /* set count to number of bytes of returned data */
|
|     rule_array_count = 2;
|
|     return_code = 0;
|     reason_code = 0;
|
|     /* set keywords in the rule array */
|
|     memcpy(rule_array2,"ADAPTER1TIMEDATE",16);
|
|     /* get the time from the card */
|
|     CSUACFQ( &return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             &rule_array_count,
|             (char *)rule_array2,
|             &verb_data_length,
|             verb_data);
|
|     if ( (return_code == OK) | (return_code == WARNING) )
|     {
| printf("Time and date was successfully returned.¥n");
| printf("Return/reason codes ");
| printf("%1d/%1d¥n¥n", return_code, reason_code);
|
| printf("DATE = %.8s¥n", rule_array2);
| printf("TIME = %.8s¥n", &rule_array2[1]);
| printf("DAY of WEEK = %.8s¥n", &rule_array2[2]);
|     }
|
|     else
|     {
| printf("An error occurred while getting the time and date.¥n");
| printf("Return/reason codes ");
| printf("%1d/%1d¥n¥n", return_code, reason_code);
|
|     return(ERROR);
|     }
| }

```

例: 暗号化コプロセッサの情報を要求する:

暗号化コプロセッサについての情報を要求するには、必要に応じて以下のプログラム例を変更してください。このプログラムでは、2番目の必須キーワードをユーザーに求めるプロンプトが表示されます。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
| ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これ
| らのプログラム例を修正して使用することをお勧めします。

```
| /*-----*/  
| /* Query the card for status or other information. */  
| /* This sample program prompts the user for the second required */  
| /* keyword. (ADAPTER1 keyword is assumed.) */  
| /* */  
| /* */  
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999 */  
| /* */  
| /* This material contains programming source code for your */  
| /* consideration. These examples have not been thoroughly */  
| /* tested under all conditions. IBM, therefore, cannot */  
| /* guarantee or imply reliability, serviceability, or function */  
| /* of these program. All programs contained herein are */  
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */  
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */  
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */  
| /* these programs and files. */  
| /* */  
| /* */  
| /* Note: This verb is more fully described in Chapter 2 of */  
| /* IBM CCA Basic Services Reference and Guide */  
| /* (SC31-8609) publication. */  
| /* */  
| /* Parameters: */  
| /* char * keyword2 upto 8 bytes */  
| /* */  
| /* Example: */  
| /* CALL PGM(CFQ) TIMEDATE */  
| /* */  
| /* */  
| /* Note: This program assumes the device to use is */  
| /* already identified either by defaulting to the CRP01 */  
| /* device or by being explicitly named using the */  
| /* Cryptographic_Resource_Allocate verb. Also this */  
| /* device must be varied on and you must be authorized */  
| /* to use this device description. */  
| /* */  
| /* Use these commands to compile this program on the system: */  
| /* ADDLIB LIB(QCCA) */  
| /* CRTCMOD MODULE(CFQ) SRCFILE(SAMPLE) */  
| /* CRTPGM PGM(CFQ) MODULE(CFQ) BNDSRVPGM(QCCA/CSUACFQ) */  
| /* */  
| /* Note: Authority to the CSUACFQ service program in the */  
| /* QCCA library is assumed. */  
| /* */  
| /* The Common Cryptographic Architecture (CCA) verb used is */  
| /* Cryptographic_Facility_Query (CSUACFQ). */  
| /* */  
| /*-----*/  
|  
| #include "csucincl.h" /* header file for CCA Cryptographic */  
| /* Service Provider */  
| #include <stdio.h>  
| #include <string.h>  
| #include <stdlib.h>  
|  
| /*-----*/  
| /* standard return codes */
```

```

| /*-----*/
| #define ERROR      -1
| #define OK         0
| #define WARNING    4
|
| int main(int argc, char *argv[])
|
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[18][8];
|     long rule_array_count = 2;
|
|     /*-----*/
|     /* fields unique to this sample program */
|     /*-----*/
|
|     long verb_data_length = 0; /* currently not used by this verb */
|     char * verb_data = " ";
|
|     int i;
|
|     /* check the keyboard input */
|
|     if (argc != 2)
|     {
| printf("You did not enter the keyword parameter.\n");
| printf("Enter one of the following:  STATCCA, STATCARD, ");
| printf("STATDIAG, STATEXPT, STATMOFN, STATEID, TIMEDATE\n");
| return(ERROR);
|     }
|
|     if ( (strlen(argv[1]) > 8) | (strlen(argv[1]) < 7) )
|     {
| printf("Your input string is not the right length.\n");
| printf("Input keyword must be 7 or 8 characters.\n");
|
|         printf("Enter one of the following:  STATCCA, STATCARD, ");
| printf("STATDIAG, STATEXPT, STATMOFN, STATEID, TIMEDATE\n");
| return(ERROR);
|     }
|
|     /* set keywords in the rule array */
|
|     memcpy(rule_array,"ADAPTER1      ",16);
|     memcpy(&rule_array[1], argv[1], strlen(argv[1]));
|
|     /* get the requested data from the card */
|
|     CSUACFQ( &return_code,
|             &reason_code,
|             &exit_data_length,

```

```

|         exit_data,
|         &rule_array_count,
|         (char *)rule_array,
|         &verb_data_length,
|         verb_data);
|
|     if ( (return_code == OK) | (return_code == WARNING) )
|     {
|
| printf("Requested data was successfully returned.\n");
|
| printf("Return/reason codes ");
|
| printf("%ld/%ld\n\n", return_code, reason_code);
|
| printf("%s data = ", argv[1]);
|
| for (i = 0; i < 8 * rule_array_count; i++)
|     printf("%c", rule_array[i / 8][i % 8]);
| printf("\n");
|     }
|
|     else
|     {
| printf("An error occurred while getting the requested data.\n");
|
| printf("You requested %s\n", argv[1]);
|
| printf("Return/reason codes ");
|
| printf("%ld/%ld\n\n", return_code, reason_code);
|
|     return(ERROR);
|     }
| }

```

鍵ストア・ファイルの初期化

鍵ストア・ファイルは、操作鍵、つまりマスター鍵で暗号化された鍵を保管するデータベース・ファイルです。DES 鍵および PKA 鍵のレコードを保持する場合は、ここに記載された情報を参照してください。

暗号化コプロセッサの 2 つの異なるタイプの鍵ストアを初期化することができます。暗号化コプロセッサは、1 つのタイプを PKA 鍵を保管するために使用し、もう 1 つのタイプを DES 鍵を保管するために使用します。鍵ストア・ファイルに鍵を保管する場合は、鍵ストア・ファイルを初期化する必要があります。鍵ストア・ファイルには保管鍵が保管されませんが、CCA はコプロセッサ内でラベルを検索する前に鍵ストア・ファイル内でラベルを検索するため、鍵ストア・ファイルが 1 つ必要です。

CCA CSP は、DB2[®] 鍵ストア・ファイルが存在していない場合には、これを作成します。鍵ストア・ファイルがすでに存在している場合には、CCA CSP はそのファイルを削除して、新しい鍵ストア・ファイルを作成します。

鍵ストアを初期化するために、暗号化コプロセッサ構成ユーティリティーを使用することができます。「構成の管理」をクリックし、次に、初期化する鍵ストア・ファイルに応じて、「DES 鍵」または「PKA 鍵」をクリックします。このユーティリティーでは、存在していなかったファイルの初期化しかできません。

独自のアプリケーションを作成して、鍵ストア・ファイルを初期化することもできます。それには、KeyStore_Initialize (CSNBKSI) API verb を使用します。

| 暗号化コプロセッサ用の鍵ストアを作成すると、DES 鍵と PKA 鍵を生成し、鍵ストア・ファイルに保管することができます。

| 関連概念

| 2 ページの『暗号化の概念』

| ここでは、暗号化機能の基本的な知識およびサーバーの暗号化サービスの概要を示します。

| 149 ページの『DES キーおよび PKA キーの作成』

| DES および PKA 鍵を作成して、DES 鍵ストアに保管することができます。

| 例: 暗号化コプロセッサ用の鍵ストアを初期化するための ILE C プログラム:

| 暗号化コプロセッサの鍵ストアを初期化するには、必要に応じて以下のプログラム例を変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
| /*-----*/
| /* Create key store files for PKA keys.          */
| /*                                              */
| /* COPYRIGHT      5769-SS1 (c) IBM Corp 1999, 2000 */
| /*                                              */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these programs. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files.                    */
| /*                                              */
| /* Parameters:                                  */
| /* Qualified File Name                          */
| /*                                              */
| /* Examples:                                    */
| /* CALL PGM(INZPKEYST) PARM('QGPL/PKAFILE')    */
| /*                                              */
| /* Use the following commands to compile this program: */
| /* ADDLIB LIB(QCCA)                             */
| /* CRTCMOD MODULE(INZPKEYST) SRCFILE(SAMPLE)   */
| /* CRTPGM PGM(INZPKEYST) MODULE(INZPKEYST) +  */
| /*      BNDSRVPGM(QCCA/CSNBKSI)               */
| /*                                              */
| /* Note: authority to the CSNBKSI service program in the */
| /*      QCCA library is assumed.                */
| /*                                              */
| /* Common Cryptographic Architecture (CCA) verbs used: */
| /* Keystore_Initialize (CSNBKSI)               */
| /*                                              */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h"          /* header file for CCA Cryptographic
|                               Service Provider          */
```

```

| int main(int argc, char *argv[])
| {
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
|
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code;
| long reason_code;
| long exit_data_length;
| char exit_data[2];
| char rule_array[4][8];
| long rule_array_count;
|
| /*-----*/
| /* fields unique to this sample program */
| /*-----*/
|
| long file_name_length;
| unsigned char description[4];
| long description_length = 0;
| unsigned char masterkey[8];
|
| /*-----*/
| /* Check if file name was passed */
| /*-----*/
|
| if(argc < 2)
| {
|     printf("File name was not specified.\n");
|     return ERROR;
| }
|
| /*-----*/
| /* fill in parameters for Keystore_Initialize */
| /*-----*/
|
| rule_array_count = 2;
| memcpy((char*)rule_array,"CURRENT PKA ",16);
| file_name_length = strlen(argv[1]);
|
| /*-----*/
| /* Create key store file */
| /*-----*/
|
| CSNBKSI(&return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (char*)rule_array,
|         &file_name_length,
|         argv[1],
|         &description_length,
|         description,
|         masterkey);
|
| /*-----*/
| /* Check the return code and display the result */
| /*-----*/
|
| if (return_code != 0)
| {

```

```

|     printf("Request failed with return/reason codes: %d/%d\n",
|           return_code, reason_code);
|     return ERROR;
|   }
|   else
|   {
|     printf("Key store file created\n");
|     return OK;
|   }
| }

```

例: 暗号化コプロセッサ用の鍵ストアを初期化するための ILE RPG プログラム:

暗号化コプロセッサの鍵ストアを初期化するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| D*****
| D* INZPKAST
| D*
| D* Create key store files for PKA keys.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D*       IBM CCA Basic Services Reference and Guide
| D*       (SC31-8609) publication.
| D*
| D* Parameters: None
| D*
| D* Example:
| D* CALL PGM(INZPKEYST) ('QGPL/PKAKEYS')
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(INZPKAST) SRCFILE(SAMPLE)
| D* CRTPGM PGM(INZPKEYST) MODULE(INZPKEYST)
| D*       BNDSRVPGM(QCCA/CSNBKSI)
| D*
| D* Note: Authority to the CSNBKSI service program in the
| D*       QCCA library is assumed.
| D*
| D* The Common Cryptographic Architecture (CCA) verbs used are
| D* Key_Store_Initialize (CSNBKSI)
| D*
| D*****
| D*-----

```



```

| D* Declare variables for CCA SAPI calls
| D*-----
| D*          ** Return code
| DRETURNCODE S          9B 0
| D*          ** Reason code
| DREASONCODE S          9B 0
| D*          ** Exit data length
| DEXITDATALEN S         9B 0
| D*          ** Exit data
| DEXITDATA S           4
| D*          ** Rule array count
| DRULEARRAYCNT S        9B 0
| D*          ** Rule array
| DRULEARRAY S          16
| D*          ** File name length
| DFILENAMELEN S         9B 0
| D*          ** File name
| DFILENAME S           21
| D*          ** Description length
| DDESCRIPLEN S          9B 0
| D*          ** Description
| DDESCRIP S            16
| D*          ** Master key part
| DMASTERKEY S          24
| D*
| D*****
| D* Prototype for Key_Store_Initialize (CSNBKSI)
| D*****
| DCSNBKSI PR
| DRETCODE          9B 0
| DRSNCODE          9B 0
| DEXTDTALEN        9B 0
| DEXTDTA           4
| DRARRAYCT         9B 0
| DRARRAY           16
| DFILENMLN         9B 0
| DFILENM           21
| DDSCPLN           9B 0
| DDSCRIP           16
| DMSTRKY           24
| D*
| D*-----
| D*          ** Declares for sending messages to the
| D*          ** job log using the QMHSNDPM API
| D*-----
| DMSG S          75 DIM(2) CTDATA PERRCD(1)
| DMSGLENGTH S     9B 0 INZ(75)
| D DS
| DMSGTEXT          1 75
| DFAILRET          41 44
| DFAILRSNC         46 49
| DMESSAGEID S      7 INZ(' ')
| DMESSAGEFILE S    21 INZ(' ')
| DMSGKEY S          4 INZ(' ')
| DMSGTYPE S        10 INZ('*INFO ')
| DSTACKENTRY S     10 INZ('* ')
| DSTACKCOUNTER S   9B 0 INZ(2)
| DERRCODE DS
| DBYTESIN          1 4B 0 INZ(0)
| DBYTESOUT         5 8B 0 INZ(0)
| D*
| C*****
| C* START OF PROGRAM *
| C*****
| C *ENTRY PLIST
| C PARM FILENAME
| C*-----

```

```

C* Set the keyword in the rule array *
C*-----*
C          MOVEL   'PKA   '   RULEARRAY
C          MOVE    'CURRENT'  RULEARRAY
C          Z-ADD   2         RULEARRAYCNT
C*-----*
C* Set the description length *
C*-----*
C          Z-ADD   0         DESCRIPLEN
C*-----*
C* Find the file name length *
C*-----*
C          EVAL    FILENAMELEN = %LEN(%TRIM(FILENAME))
C*****
C* Call Key Store Initialize SAPI *
C*****
C          CALLP   CSNBKSI   (RETURNCODE:
C                          REASONCODE:
C                          EXITDATALEN:
C                          EXITDATA:
C                          RULEARRAYCNT:
C                          RULEARRAY:
C                          FILENAMELEN:
C                          FILENAME:
C                          DESCRIPLEN:
C                          DESCRIP:
C                          MASTERKEY)
C* *-----*
C* * Check the return code *
C* *-----*
C          RETURNCODE  IFGT    4
C* *-----*
C* * Send failure message *
C* *-----*
C          MOVEL   MSG(1)     MSGTEXT
C          MOVE    RETURNCODE  FAILRETC
C          MOVE    REASONCODE  FAILRSNC
C          EXSR    SNDMSG
C          RETURN
C          ENDIF
C*
C* *-----*
C* * Send success message *
C* *-----*
C          MOVEL   MSG(2)     MSGTEXT
C          EXSR    SNDMSG
C*
C          SETON                                     LR
C*
C*****
C* Subroutine to send a message
C*****
C          SNDMSG   BEGSR
C          CALL     'QMHSNDPM'
C          PARM     MESSAGEID
C          PARM     MESSAGEFILE
C          PARM     MSGTEXT
C          PARM     MSGLENGTH
C          PARM     MSGTYPE
C          PARM     STACKENTRY
C          PARM     STACKCOUNTER
C          PARM     MSGKEY
C          PARM     ERRCODE
C          ENDSR

```

```

**
| CSNBKSI failed with return/reason codes 9999/9999.
| The file was succesully initialized.

```

DES キーおよび PKA キーの作成

DES および PKA 鍵を作成して、DES 鍵ストアに保管することができます。

暗号化コプロセッサを使用すると 2 つのタイプの暗号鍵を作成することができます。

- データ暗号化規格 (DES) 鍵の内容は、対称アルゴリズムに基づいています。したがって、暗号はデータの暗号化および暗号化解除を行うために同じ鍵値を使用します。DES 鍵は、ファイルの暗号化や暗号化解除、PINS の操作、および鍵の管理に使用します。

暗号化コプロセッサで DES 鍵を作成するには、プログラムを作成します。

- 公開鍵アルゴリズム (PKA) 鍵の内容は非対称アルゴリズムに基づいているので、暗号は暗号化と暗号化解除両方に違った鍵を使用します。デジタル・シグニチャーを使用したファイルの署名、および鍵の管理には、PKA 鍵を使用します。

暗号化コプロセッサで PKA 鍵を作成するには、プログラムを作成します。

注: 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

DES 鍵と PKA 鍵は、鍵ストア・ファイルを使用して、それらの鍵用に作成した鍵ストア・ファイルに保管します。暗号化コプロセッサに PKA 鍵を保管することもできます。ハードウェアに鍵を保管する方法については、<http://www.ibm.com/security/cryptocards/library.shtml> の情報を参照してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

関連概念

127 ページの『暗号化コプロセッサのログオンまたはログオフ』
暗号化コプロセッサにログオンまたはログオフするには、役割制限のある API を使用します。

175 ページの『デジタル署名の生成および検査』
デジタル署名と呼ばれる識別値の検査を組み込んでおくことにより、検知されない変更からデータを保護することができます。

143 ページの『鍵ストア・ファイルの初期化』
鍵ストア・ファイルは、操作鍵、つまりマスター鍵で暗号化された鍵を保管する データベース・ファイルです。DES 鍵および PKA 鍵のレコードを保持する場合は、ここに記載された情報を参照してください。

関連タスク

162 ページの『PIN の処理』
金融機関は、その顧客に対して個人金融取引を許可するために個人識別番号 (PIN) を使用します。PIN はパスワードに似ていますが、10 進数で構成されており、通常、関連したアカウント番号の暗号化関数です。暗号化コプロセッサを使用すると、PIN の処理を行うことができます。

関連情報

ファイルの暗号化または暗号化解除
暗号化コプロセッサのさらに実用的な用途の 1 つとして、データ・ファイルの暗号化と暗号化解除があります。

例: 暗号化コプロセッサで DES 鍵を作成する:

暗号化コプロセッサで DES 鍵を作成するには、必要に応じて以下のプログラム例を変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
| ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これ
| らのプログラム例を修正して使用することをお勧めします。

```
| /*-----*/  
| /* Generate DES keys in key store. */  
| /* */  
| /* COPYRIGHT 5769-SS1 (c) IBM Corp 1999 */  
| /* */  
| /* This material contains programming source code for your */  
| /* consideration. These examples have not been thoroughly */  
| /* tested under all conditions. IBM, therefore, cannot */  
| /* guarantee or imply reliability, serviceability, or function */  
| /* of these programs. All programs contained herein are */  
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */  
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */  
| /* EXPRESSLY DISCLAIMED. IBM provides no program services for */  
| /* these programs and files. */  
| /* */  
| /* Parameters: */  
| /* char * key label, 1 to 64 characters */  
| /* char * key store name, 1 to 21 characters in form 'lib/file' */  
| /* (optional, see second note below) */  
| /* */  
| /* Examples: */  
| /* CALL PGM(KEYGEN) PARM('TEST.LABEL.1') */  
| /* */  
| /* CALL PGM(KEYGEN) PARM('MY.OWN.LABEL' 'QGPL/MYKEYSTORE') */  
| /* */  
| /* Note: This program assumes the device you want to use is */  
| /* already identified either by defaulting to the CRP01 */  
| /* device or has been explicitly named using the */  
| /* Cryptographic_Resource_Allocate verb. Also this */  
| /* device must be varied on and you must be authorized */  
| /* to use this device description. */  
| /* */  
| /* If the key store name parameter is not provided, this */  
| /* program assumes the key store file you will use is */  
| /* already identified either by being specified on the */  
| /* cryptographic device or has been previously named */  
| /* using the Key_Store_Designate verb. Also you must be */  
| /* authorized to add and update records in this file. */  
| /* */  
| /* Use the following commands to compile this program: */  
| /* ADDLIB LIB(QCCA) */  
| /* CRTCMOD MODULE(KEYGEN) SRCFILE(SAMPLE) */  
| /* CRTPGM PGM(KEYGEN) MODULE(KEYGEN) + */  
| /* BNDSRVPGM(QCCA/CSUAKSD QCCA/CSNBKRC QCCA/CSNBKGN) */  
| /* */  
| /* Note: authority to the CSUAKSD, CSNBKRC and CSNBKGN service */  
| /* programs in the QCCA library is assumed. */  
| /* */  
| /* Common Cryptographic Architecture (CCA) verbs used: */  
| /* Key_Store_Designate (CSUAKSD) */  
| /* DES_Key_Record_Create (CSNBKRC) */  
| /* Key_Generate (CSNBKGN) */  
| /* */  
| /*-----*/  
|  
| #include <stdlib.h>  
| #include <stdio.h>  
| #include <string.h>  
| #include "csucincl.h" /* header file for CCA Cryptographic
```

```

|                                     Service Provider                                     */
|
| int main(int argc, char *argv[])
| {
|
| /*-----*/
| /* standard return codes                                     */
| /*-----*/
|
| #define ERROR -1
| #define OK    0
|
| /*-----*/
| /* standard CCA parameters                                 */
| /*-----*/
|
|     long return_code;
|     long reason_code;
|     long exit_data_length;
|     char exit_data[2];
|     long rule_array_count;
|
| /*-----*/
| /* fields unique to this sample program                   */
| /*-----*/
|
|     long file_name_length;
|     char key_label[64];
|
| /*-----*/
| /* See if the user wants to specify which key store file to use */
| /*-----*/
|
|     if(argc > 2)
|     {
|         file_name_length = strlen(argv[2]);
|
|         if((file_name_length > 0) &&
| (file_name_length < 22))
|         {
|             rule_array_count = 1;
|
|             CSUAKSD(&return_code,
| &reason_code,
| &exit_data_length,
| exit_data,
| &rule_array_count,
| "DES ", /* rule_array, we are working with
|         DES keys in this sample program */
| &file_name_length,
| argv[2]); /* key store file name */
|
|             if (return_code != 0)
|             {
|                 printf("Key store designate failed for reason %d/%d\n\n",
| return_code, reason_code);
|                 return ERROR;
|             }
|             else
|             {
|                 printf("Key store designated\n");
|                 printf("SAPI returned %ld/%ld\n", return_code, reason_code);
|             }
|             }
|             else
|             {
|                 printf("Key store file name is wrong length");

```

```

|     return ERROR;
|     }
| }
| else;                               /* let key store file name default */
|
| /*-----*/
| /* Create a record in key store      */
| /*-----*/
|
| memset(key_label, ' ', 64);
| memcpy(key_label, argv[1], strlen(argv[1]));
|
| CSNBKRC(&return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         key_label);
|
| if (return_code != 0)
| {
|     printf("Record could not be added to key store for reason %d/%d\n",
|            return_code, reason_code);
|     return ERROR;
| }
| else
| {
|     printf("Record added to key store\n");
|     printf("SAPI returned %ld/%ld\n", return_code, reason_code);
| }
|
| /*-----*/
| /* Generate a key                    */
| /*-----*/
|
| CSNBKGN(&return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         "OP ", /* operational key is requested */
|         "SINGLE ", /* single length key requested */
|         "DATA ", /* Data encrypting key requested */
|         " ", /* second value must be blanks when
| key form requests only one key */
|         "¥0", /* key encrypting key is null for
| operational keys */
|         "¥0", /* key encrypting key is null since
| only one key is being requested */
|         key_label, /* store generated key in key store*/
|         "¥0"); /* no second key is requested */
|
| if (return_code != 0)
| {
|     printf("Key generation failed for reason %d/%d\n",
|            return_code, reason_code);
|     return ERROR;
| }
| else
| {
|     printf("Key generated and stored in key store\n");
|     printf("SAPI returned %ld/%ld\n", return_code, reason_code);
|     return OK;
| }
| }
|

```

例: 暗号化コプロセッサーで PKA 鍵を作成する:

暗号化コプロセッサーで PKA 鍵を作成するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
/*-----*/
/* Generate PKA keys in key store.                */
/*                                               */
/* COPYRIGHT      5769-SS1 (c) IBM Corp 1999    */
/*                                               */
/* This material contains programming source code for your
/* consideration. These examples have not been thoroughly
/* tested under all conditions. IBM, therefore, cannot
/* guarantee or imply reliability, serviceability, or function
/* of these programs. All programs contained herein are
/* provided to you "AS IS". THE IMPLIED WARRANTIES OF
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
/* EXPRESSLY DISCLAIMED. IBM provides no program services for
/* these programs and files.
/*
/* Parameters:
/* char * key label, 1 to 64 characters
/*
/* Examples:
/* CALL PGM(PKAKEYGEN) PARM('TEST.LABEL.1')
/*
/* Note: This program assumes the card you want to load is
/* already identified either by defaulting to the CRP01
/* device or has been explicitly named using the
/* Cryptographic_Resource_Allocate verb. Also this
/* device must be varied on and you must be authorized
/* to use this device description.
/*
/* This program also assumes the key store file you will
/* use is already identified either by being specified on
/* the cryptographic device or has been explicitly named
/* using the Key_Store_Designate verb. Also you must be
/* authorized to add and update records in this file.
/*
/* Use the following commands to compile this program:
/* ADDLIB LIB(QCCA)
/* CRTCMOD MODULE(PKAKEYGEN) SRCFILE(SAMPLE)
/* CRTPGM PGM(PKAKEYGEN) MODULE(PKAKEYGEN) +
/* BNSRVPGM(QCCA/CSNDKRC QCCA/CSNDPKG)
/*
/* Note: authority to the CSNDKRC and CSNDPKG service programs
/* in the QCCA library is assumed.
/*
/* Common Cryptographic Architecture (CCA) verbs used:
/* PKA_Key_Record_Create (CSNDKRC)
/* PKA_Key_Generate (CSNDPKG)
/*-----*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "csucincl.h"          /* header file for CCA Cryptographic
                               Service Provider */
```

```

| int main(int argc, char *argv[])
| {
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
|
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code;
| long reason_code;
| long exit_data_length;
| char exit_data[2];
| char rule_array[4][8];
| long rule_array_count;
|
| /*-----*/
| /* fields unique to this sample program */
| /*-----*/
|
| char key_label[64]; /* identify record in key store to
| hold generated key */
| #pragma pack (1)
|
| typedef struct rsa_key_token_header_section {
| char token_identifier;
| char version;
| short key_token_struct_length;
| char reserved_1[4];
| } rsa_key_token_header_section;
|
| typedef struct rsa_private_key_1024_bit_section {
| char section_identifier;
| char version;
| short section_length;
| char hash_of_private_key[20];
| short reserved_1;
| short master_key_verification_pattern;
| char key_format_and_security;
| char reserved_2;
| char hash_of_key_name[20];
| char key_usage_flag;
| char rest_of_private_key[312];
| } rsa_private_key_1024_bit_section;
|
| typedef struct rsa_public_key_section {
| char section_identifier;
| char version;
| short section_length;
| short reserved_1;
| short exponent_field_length;
| short modulus_length;
| short modulus_length_in_bytes;
| char exponent;
| } rsa_public_key_section;
|
| struct {
| rsa_key_token_header_section rsa_header;
| rsa_private_key_1024_bit_section rsa_private_key;
| rsa_public_key_section rsa_public_key;
| } key_token;

```



```

| struct {
|     short modlen;
|     short modlenfld;
|     short pubexplen;
|     short prvexplen;
|     long pubexp;
| } prvPubl;
|
| #pragma pack ()
|
|     long key_struct_length;
|     long zero = 0;
|     long key_token_length;
|
|     long regen_data_length;
|     long generated_key_id_length;
|
| /*-----*/
| /* Create record in key store */
| /*-----*/
|     rule_array_count = 0;
|     key_token_length = 0;
|     memset(key_label, ' ', 64);
|     memcpy(key_label, argv[1], strlen(argv[1]));
|
|     CSNDKRC(&return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             &rule_array_count,
|             "¥0", /* rule_array */
|             key_label,
|             &key_token_length,
|             "¥0"); /* key token */
|
|     if (return_code != 0)
|     {
|         printf("Record could not be added to key store for reason %d/%d¥n¥n",
|               return_code, reason_code);
|         return ERROR;
|     }
|     else
|     {
|         printf("Record added to key store¥n");
|         printf("SAPI returned %ld/%ld¥n", return_code, reason_code);
|     }
|
| /*-----*/
| /* Build a key token, needed to generate PKA key */
| /*-----*/
|     memset(&key_token, 0X00, sizeof(key_token));
|
|     key_token.rsa_header.token_identifier = 0X1E; /* external token */
|     key_token.rsa_header.key_token_struct_length = sizeof(key_token);
|
|     key_token.rsa_private_key.section_identifier =
|         0X02; /* RSA private key */
|     key_token.rsa_private_key.section_length =
|         sizeof(rsa_private_key_1024_bit_section);
|     key_token.rsa_private_key.key_usage_flag = 0X80;
|
|     key_token.rsa_public_key.section_identifer = 0X04; /* RSA public key */
|     key_token.rsa_public_key.section_length =
|         sizeof(rsa_public_key_section);
|     key_token.rsa_public_key.exponent_field_length = 1;

```

```

| key_token.rsa_public_key.modulus_length = 512;
| key_token.rsa_public_key.exponent = 0x03;
|
| key_token_length = sizeof(key_token);
|
| printf("Key token built\n");
|
| /*-----*/
| /* Generate a key */
| /*-----*/
|
| rule_array_count = 1;
| regen_data_length = 0;
| /* key_token_length = 64; */
| generated_key_id_length = 2500;
|
| CSNDPKG(&return_code,
| &reason_code,
| &exit_data_length,
| exit_data,
| &rule_array_count,
| "MASTER ", /* rule_array */
| &regen_data_length,
| "¥0", /* regeneration_data, none needed */
| &key_token_length, /* skeleton_key_token_length */
| (char *)&key_token, /* skeleton_key_token built above */
| "¥0", /* transport_id, only needed for
| XPORT keys */
| &generated_key_id_length,
| key_label); /* generated_key_id, store generated
| key in key store */
|
| if (return_code != 0)
| {
| printf("Key generation failed for reason %d/%d\n\n",
| return_code, reason_code);
| return ERROR;
| }
| else
| {
| printf("Key generated and stored in key store\n");
| printf("SAPI returned %d/%d\n\n", return_code, reason_code);
| return OK;
| }
| }

```

ファイルの暗号化または暗号化解除

暗号化コプロセッサのさらに実用的な用途の 1 つとして、データ・ファイルの暗号化と暗号化解除があります。

これらの暗号方法の 1 つを使用してファイルを保護することができます。

- ファイル全体をバイトの文字列として処理する (この方法は、プログラム例が使用している方法です)。
- 各レコードまたは各レコードの部分を暗号化する。

独自のプログラムを作成して、データ・ファイルだけではなく多くの異なる形式のデータを保護します。

例: データを暗号化コプロセッサで暗号化する:

暗号化コプロセッサでデータを暗号化するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

| 付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリ
 | ティー上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これ
 | らのプログラム例を修正して使用することをお勧めします。

```

| /*-----*/
| /* */
| /* Sample C program for enciphering data in a file. */
| /* */
| /* COPYRIGHT      5769-SS1 (c) IBM Corp 1999      */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these programs. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* Parameters: */
| /* char * key label, 1 to 64 characters */
| /* char * input file name, 1 to 21 characters (lib/file) */
| /* char * output file name, 1 to 21 characters (lib/file) */
| /* */
| /* Example: */
| /* CALL PGM(ENCFILE) PARM( 'MY.KEY.LABEL' 'QGPL/MYDATA' + */
| /* 'QGPL/CRYPTDATA' ) */
| /* */
| /* Note: This program assumes the device you want to use is */
| /* already identified either by defaulting to the CRP01 */
| /* device or has been explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* This program assumes the key store file you will use is */
| /* already identified either by being specified on the */
| /* cryptographic device or has been previously named */
| /* using the Key_Store_Designate verb. Also you must be */
| /* authorized to add and update records in this file. */
| /* */
| /* The output file should NOT have key fields since all */
| /* data in the file will be encrypted and therefore trying */
| /* to sort the data will be meaningless. */
| /* (This is NOT checked by the program) */
| /* */
| /* Use the following commands to compile this program: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(ENCFILE) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(ENCFILE) MODULE(ENCFILE) + */
| /* BNSRVPGM(QCCA/CSNBENC) */
| /* */
| /* Note: authority to the CSNBENC service program in the */
| /* QCCA library is assumed. */
| /* */
| /* Common Cryptographic Architecture (CCA) verbs used: */
| /* Encipher (CSNBENC) */
| /* */
| /*-----*/
|
| /*-----*/
| /* Retrieve various structures/utilities that are used in program. */
| /*-----*/
|
| #include <stdio.h> /* Standard I/O header. */

```

```

| #include <stdlib.h>           /* General utilities.          */
| #include <stddef.h>         /* Standard definitions.       */
| #include <string.h>         /* String handling utilities.   */
| #include "csucincl.h"       /* header file for CCA Cryptographic
|                             Service Provider
|
|
| /*-----*/
| /* Declares for working with files.
| /*-----*/
| #include <xxfdbk.h>         /* Feedback area structures.   */
| #include <recio.h>         /* Record I/O routines        */
| _RFILE *dbfptr;           /* Pointer to database file.    */
| _RFILE *dbfptre;          /* Pointer to database file.    */
| _RIOFB_T *db_fdbk;        /* I/O Feedback - data base file
| _XXOPFB_T *db_opfb;
| _XXOPFB_T *db_opfbe;
|
| /*-----*/
| /* Declares for working with user space objects.
| /*-----*/
| #include "qusptrus.h"
| #include "quscrtus.h"
| #include "qusdltus.h"
| #define USSPC_ATTR          "PF          "
| #define USSPC_INIT_VAL     0x40
| #define USSPC_AUTH         "*EXCLUDE "
| #define USSPC_TEXT         "Sample user space"
| #define USSPC_REPLACE      "*YES          "
|
| char space_name[21] = "PLAINTXT QTEMP "; /* Name of user
|                                     space for plain text
| char cipher_name[21] = "CIPHER QTEMP "; /* Name for user
|                                     space containing ciphertext
|
| struct {
|     /* Error code structure required for
|     /* the User Space API's.
|     int in_len; /* the length of the error code.
|     int out_len; /* the length of the exception data.
|     char excp_id[7]; /* the Exception ID.
|     char rev; /* Reserved Field.
|     char excp_data[120]; /* the output data associated
| } error_code; /* the exception ID.
|
| char ext_atr[11] = USSPC_ATTR; /* Space attribute
| char initial_val = USSPC_INIT_VAL;
| /* Space initial value
| char auth[11] = USSPC_AUTH; /* Space authority
| char desc[51] = USSPC_TEXT; /* Space text
| char replace[11] = USSPC_REPLACE;
| /*Space replace attribute*/
|
| /*-----*/
| /* Start of mainline code.
| /*-----*/
| int main(int argc, char *argv[])
| {
|
| /*-----*/
| /* standard return codes
| /*-----*/
|
| #define ERROR -1
| #define OK 0

```

```

| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code;
| long reason_code;
| long exit_data_length;
| char exit_data[2];
| long rule_array_count;
|
| char *user_space_ptr;
| char *user_space;
| char *cipher_spc;
| long file_bytes;
| long i;
| long j;
| char key_label[64];
|
| long text_len, pad_character;
| char initial_vector[8];
| char chaining_vector[18];
|
| /*-----*/
| /* Open database files. */
| /*-----*/
|
| if (argc < 4) /* were the correct number
| of parameters passed? */
|
| {
| printf("This program needs 3 parameters - ");
| printf("key label, input file name, output file name\n");
|
| return ERROR;
| }
|
| else
| {
|
| file_bytes = 0; /* Set initial number of
| bytes to encipher to 0 */
|
| /* Open the input file. If the file pointer, dbfptr is not
| NULL, then the file was successfully opened. */
|
| if (( dbfptr = _Ropen(argv[2], "rr riofb=n"))
| != NULL)
| {
|
| /*-----*/
| /* Determine the number of bytes that will be enciphered. */
| /*-----*/
| db_opfb = _Ropnfbk( dbfptr ); /* Get pointer to the File
| open feedback area. */
|
| file_bytes = db_opfb->num_records *
| db_opfb->pgm_record_len
| + 1; /* 1 is added to prevent an
| end of space error */
|
| j = db_opfb->num_records; /* Save number of records*/
| /*-----*/
| /* Create user space and get pointer to it. */
| /*-----*/
| error_code.in_len = 136; /* Set length of error
| structure. */
| QUSDLTUS(space_name,&error_code); /* Delete the user space

```

```

                                if it already exists. */

/* Create the plaintext user space object */
QUSCRTUS(space_name,ext_atr,file_bytes,
          &initial_val,auth,
          desc, replace,&error_code);

error_code.in_len = 48;          /* Set length of error
                                structure */

QUSPTRUS(space_name,           /* Retrieve a pointer to */
          (void *)&user_space, /* the user space. */
          (char*)&error_code);

user_space_ptr = user_space;    /* Make copy of pointer */

error_code.in_len = 136;        /* Set length of error */
                                /* structure. */
QUSDLTUS(cipher_name,&error_code); /* Delete cipher space
                                if already exists. */

/* Create ciphertext user space object */
QUSCRTUS(cipher_name,ext_atr,
          file_bytes,&initial_val,auth,
          desc, replace,&error_code);

error_code.in_len = 48;          /* Set length of error */
                                /* structure */
QUSPTRUS(cipher_name,         /* Retrieve pointer to */
          (void *)&cipher_spc, /* ciphertext user space */
          (char*)&error_code);

/*-----*/
/* Read file and fill space */
/*-----*/
for (i=1; i<=j; i++)            /* Repeat for each record */
{
    /* Read a record and place in user space. */
    db_fdbk = _Rreadn(dbfptr, user_space_ptr,
                     db_opfb->pgm_record_len, __DFT);

    /* Move the user space ahead the length of a record */
    user_space_ptr = user_space_ptr +
                     db_opfb->pgm_record_len;
}

if (dbfptr != NULL)            /* Close the file. */
    _Rclose(dbfptr);
/*-----*/
/* Encrypt data in space */
/*-----*/

memset((char *)key_label,' ',64); /* Initialize key label
                                to all blanks. */
memcpy((char *)key_label,        /* Copy key label parm */
       argv[1],strlen(argv[1]));

text_len = file_bytes - 1;
rule_array_count = 1;
pad_character = 40;
exit_data_length = 0;
memset((char *)initial_vector,'#0',8);

/* Encipher data in ciphertext user space */
CSNBENC(&return_code,
        &reason_code,
        &exit_data_length,

```

```

        exit_data,
        key_label,
        &text_len,
        user_space,
        initial_vector,
        &rule_array_count,
        "CBC", /* rule_array */
        &pad_character,
        chaining_vector,
        cipher_spc );

    if (return_code == 0) {
/*-----*/
/* Open output file */
/*-----*/
        if (( dbfptre = _Ropen(argv[3],
                             "wr riofb=n")) != NULL)
        {
            db_opfbe = _Ropnfbk( dbfptr ); /* Get pointer to
                                           the File open feedback
                                           area. */
            if(text_len % db_opfbe->pgm_record_len != 0)
            {
                printf("encrypted data will not fit into ");
                printf("an even number of records\n");

                if (dbfptre != NULL) /* Close the file. */
                    _Rclose(dbfptre);

                /*-----*/
                /* Delete both user spaces. */
                /*-----*/
                error_code.in_len = 136; /* Set length of
                                           error structure. */
                QUSDLTUS(space_name,&error_code); /* Delete the
                                                    user space */
                QUSDLTUS(cipher_name,&error_code); /* Delete
                                                    ciphertext space */

                return ERROR;
            }

/*-----*/
/* Write data from space to file. */
/*-----*/
            user_space_ptr = cipher_spc; /* Save pointer to
                                           cipher space. */

            j = text_len / db_opfbe->pgm_record_len; /* find
                how many records
                are needed to store
                result in output

                file */
            for (i=1; i<=j; i++) /* Repeat for each
                                   record */
            {
                /* Write data to output file */
                db_fdbk = _Rwrite(dbfptre, user_space_ptr,
                                   db_opfbe->pgm_record_len);

                /* Advance pointer ahead the length of a record */
                user_space_ptr = user_space_ptr +
                    db_opfbe->pgm_record_len;
            }
            if (dbfptre != NULL) /* Close the file */
                _Rclose(dbfptre);

```

```

        }
        /* end of open open
        output file */
    else
    {
        printf("Output file %s could not be opened\n",
        argv[3]);

        /*-----*/
        /* Delete both user spaces. */
        /*-----*/
        error_code.in_len = 136; /* Set length of
        error structure. */
        QUSDLTUS(space_name,&error_code); /* Delete the
        user space */
        QUSDLTUS(cipher_name,&error_code); /* Delete
        ciphertext space */
        return ERROR;
    }

    } /* If return code = 0 */
    else
    {
        printf("Bad return/reason code : %d/%d %n",
        return_code,reason_code);
        /*-----*/
        /* Delete both user spaces. */
        /*-----*/
        error_code.in_len = 136; /* Set length of
        error structure. */
        QUSDLTUS(space_name,&error_code); /* Delete the
        user space */
        QUSDLTUS(cipher_name,&error_code); /* Delete
        ciphertext space */
        return ERROR;
    }

    /*-----*/
    /* Delete both user spaces. */
    /*-----*/
        error_code.in_len = 136; /* Set length of
        error structure. */
        QUSDLTUS(space_name,&error_code); /* Delete the user
        space */
        QUSDLTUS(cipher_name,&error_code); /* Delete ciphertext
        space */

    } /* End of open
    input file */
    else
    {
        printf("Input file %s could not be opened\n", argv[2]);
        return ERROR;
    }
    } /* argv[] == null */
    return OK;
}

```

PIN の処理

金融機関は、その顧客に対して個人金融取引を許可するために個人識別番号 (PIN) を使用します。PIN はパスワードに似ていますが、10 進数で構成されており、通常、関連したアカウント番号の暗号化関数です。暗号化コプロセッサを使用すると、PIN の処理を行うことができます。

PIN を処理するには、プログラムを作成します。

関連概念

149 ページの『DES キーおよび PKA キーの作成』
DES および PKA 鍵を作成して、DES 鍵ストアに保管することができます。

例: 暗号化コプロセッサ上での PIN の処理:

暗号化コプロセッサで PIN の処理を行うには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
F*****
F* PINSAMPLE
F*
F* Sample program that shows the use of the appropriate
F* CCA Security API (SAPI) verbs for generating and verifying
F* PINS
F*
F* The keys are created by first building a key token
F* and then importing key parts using Key_Part_Import.
F* Four keys are created each with a different
F* key type - PINGEN, PINVER, IPINENC, and OPINENC. The
F* PINGEN key will be used to generate a Clear PIN with the
F* Clear_PIN_Generate verb. The OPINENC key will be used
F* to encrypt the PIN with the Clear_PIN_Encrypt verb.
F* The Encrypted_PIN_Verify with verify that the PIN is good
F* using the IPINENC key (to decrypt) and the PINVER key
F* to verify the PIN.
F*
F* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999
F*
F* This material contains programming source code for your
F* consideration. These example has not been thoroughly
F* tested under all conditions. IBM, therefore, cannot
F* guarantee or imply reliability, serviceability, or function
F* of these programs. All programs contained herein are
F* provided to you "AS IS". THE IMPLIED WARRANTIES OF
F* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
F* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
F* these programs and files.
F*
F*
F* Note: Input format is more fully described in Chapter 2 of
F* IBM CCA Basic Services Reference and Guide
F* (SC31-8609) publication.
F*
F* Parameters:
F* none.
F*
F* Example:
F* CALL PGM(PINSAMPLE)
F*
F* Use these commands to compile this program on the system:
F* CRTRPGMOD MODULE(PINSAMPLE) SRCFILE(SAMPLE)
F* CRTPGM PGM(PINSAMPLE) MODULE(PINSAMPLE)
F* BNDSRVPGM(QCCA/CSNBKPI QCCA/CSNBPGN +
F* QCCA/CSNBCPE QCCA/CSNBPVR)
F*
F* Note: Authority to the CSNBKPI, CSNBPGN, CSNBCPE, and
F* CSNBPVR service programs in the QCCA library is assumed.
F*
```

```

| F* The Common Cryptographic Architecture (CCA) verbs used are
| F* Key_Part_Import (CSNBKPI), Clear_PIN_Generate (CSNBPGN),
| F* Clear_PIN_Encrypt (CSNBCPE), and Encrypted_PIN_Verify (CSNBPVR).
| F*
| F* Note: This program assumes the card you want to load is
| F* already identified either by defaulting to the CRP01
| F* device or has been explicitly named using the
| F* Cryptographic_Resource_Allocate verb. Also this
| F* device must be varied on and you must be authorized
| F* to use this device description.
| F*
| F*****
| F* Declare parameters that are common to all of the CCA verbs
| F*
| F*****
| DRETURNCODE      S          9B 0
| DREASONCODE      S          9B 0
| DEXITDATALEN     S          9B 0
| DEXITDATA        S           4
| DRULEARRAYCNT    S          9B 0
| DRULEARRAY       S          16
| D*
| D*****
| D* Declare Key tokens used by this program
| D*
| D*****
| DIPINKEY         S          64
| DOPINKEY         S          64
| DPINGENKEY       S          64
| DPINVERKEY       S          64
| DKEYTOKEN        DS
| DKEYFORM         1          1
| DKEYVERSION      5          5
| DKEYFLAG1        7          7
| DKEYVALUE        17         32
| DKEYCV           33         48
| DKEYTVV          61         64B 0
| DTOKENPART1     1          16
| DTOKENPART2     17         32
| DTOKENPART3     33         48
| DTOKENPART4     49         64
| DKEYTVV1         1          4B 0
| DKEYTVV2         5          8B 0
| DKEYTVV3         9         12B 0
| DKEYTVV4         13         16B 0
| DKEYTVV5         17         20B 0
| DKEYTVV6         21         24B 0
| DKEYTVV7         25         28B 0
| DKEYTVV8         29         32B 0
| DKEYTVV9         33         36B 0
| DKEYTVV10        37         40B 0
| DKEYTVV11        41         44B 0
| DKEYTVV12        45         48B 0
| DKEYTVV13        49         52B 0
| DKEYTVV14        53         56B 0
| DKEYTVV15        57         60B 0
| D*
| D*****
| D* Declare parameters unique to Key_Part_Import
| D*
| D*****
| DCLEARKEY        S          16
| D*
| D*****
| D* Declare parameters unique to Clear_PIN_Generate,
| D* Clear_PIN_Encrypt, and Encrypted_PIN_Verify
| D*****

```

```

| DPINLEN          S          9B 0
| DPINCKL         S          9B 0
| DSEQNUMBER      S          9B 0
| DCPIN           S          16
| DEPIN           S          16
| DPAN            S          12
| DDATAARRAY      DS
| DDECTABLE       1          16
| DVALDATA        17         32
| DCLRPIN         33         48
| DPROFILE        DS
| DPINFORMAT      1          8
| DFORMATCONTROL  9          16
| DPADDIGIT       17         24
| D*
| D*****
| D* Declare variables used for creating a control vector and
| D* clear key.
| D*****
| DBLDKEY         DS
| DLEFTHALF       1          8
| DLEFTHALFA      1          4B 0
| DLEFTHALFB      5          8B 0
| DRIGHTHALF      9          16
| D*
| D*
| D*****
| D* Prototype for Key Part Import (CSNBKPI)
| D*****
| DCSNBKPI        PR
| DRETCODE        9B 0
| DRSNCODE        9B 0
| DEXTDTALEN      9B 0
| DEXTDTA         4
| DRARRAYCT       9B 0
| DRARRAY         16
| DCLRKEY         16
| DIMPKEY         64
| D*
| D*****
| D* Prototype for Clear PIN Generate (CSNBPGN)
| D*****
| DCSNBPGN        PR
| DRETCODE        9B 0
| DRSNCODE        9B 0
| DEXTDTALEN      9B 0
| DEXTDTA         4
| DPINGEN         64
| DRARRAYCT       9B 0
| DRARRAY         16
| DPINL           9B 0
| DPINCHKLEN      9B 0
| DDTAARRY        48
| DRESULT         16
| D*
| D*****
| D* Prototype for Clear PIN Encrypt (CSNBCPE)
| D*****
| DCSNBCPE        PR
| DRETCODE        9B 0
| DRSNCODE        9B 0
| DEXTDTALEN      9B 0
| DEXTDTA         4
| DPINENC         64
| DRARRAYCT       9B 0
| DRARRAY         16
| DCLRPIN         16

```

```

| DPINPROFILE          24
| DPANDATA             12
| DSEQN                9B 0
| DEPINBLCK           8
| D*
| D*****
| D* Prototype for Encrypted PIN Verify (CSNBPVR)
| D*****
| DCSNBPVR            PR
| DRETCODE             9B 0
| DRNSCODE             9B 0
| DEXTDTALEN          9B 0
| DEXTDTA              4
| DPINENC              64
| DPINVER              64
| DPINPROFILE         24
| DPANDATA             12
| DEPINBLCK           8
| DRARRAYCT           9B 0
| DRARRAY              16
| DCHECKLEN           9B 0
| DDTAARRAY           24
| D*
| D*****
| D* Declares for sending messages to job log
| D*****
| DFAILMESSAGE        S          50
| DGOODMESSAGE        S          50
| DFAILMSG            DS
| DFAILMSGTEXT        1          50
| DFAILRETC           41         44
| DFAILRSNC           46         49
| DRETSTRUCT          DS
| DRETCODE            1          4I 0
| DSLASH              5          5 INZ('/')
| DRNSCODE            6          9I 0
| DFAILMSGLENGTH      S          9B 0 INZ(49)
| DGOODMSGLENGTH      S          9B 0 INZ(29)
| DMESSAGEID          S          7          INZ(' ')
| DMESSAGEFILE        S          21         INZ(' ')
| DMSGKEY              S          4          INZ(' ')
| DMSGTYPE            S          10         INZ('*INFO ')
| DSTACKENTRY         S          10         INZ('* ')
| DSTACKCOUNTER       S          9B 0 INZ(2)
| DERRCODE            DS
| DBYTESIN            1          4B 0 INZ(0)
| DBYTESOUT           5          8B 0 INZ(0)
| C                    EVAL      FAILMESSAGE = '***** failed with return+
| C                    /reason codes 9999/9999'
| C                    EVAL      GOODMESSAGE = 'PIN Validation was successful'
| C*****
| C* START OF PROGRAM
| C*
| C*****
| C* Build a PINGEN key token
| C*
| C*****
| C* Zero out the key token to start with
| C*
| C          Z-ADD      0          KEYTVV1
| C          Z-ADD      0          KEYTVV2
| C          Z-ADD      0          KEYTVV3
| C          Z-ADD      0          KEYTVV4
| C          MOVE      TOKENPART1  TOKENPART2
| C          MOVE      TOKENPART1  TOKENPART3
| C          MOVE      TOKENPART1  TOKENPART4
| C*

```

```

| C* Set the form, version, and flag byte
| C*
| C          BITON    '7'          KEYFORM
| C          BITON    '67'         KEYVERSION
| C          BITON    '1'          KEYFLAG1
| C*
| C* The control vector for a PINGEN key that has the key part
| C* flag set is (in hex):
| C*
| C*      00227E00  03480000  00227E00  03280000
| C*
| C* If each 4 byte hex part is converted to decimal you get:
| C*
| C*      2260480   55050240  2260480   52953088
| C*
| C* Build the control vector by placing the decimal number in
| C* the appropriate half of the control vector field.
| C*****
| C          Z-ADD    2260480      LEFTHALFA
| C          Z-ADD    55050240     LEFTHALFB
| C          MOVE    LEFTHALF      KEYCV
| C          Z-ADD    2260480      LEFTHALFA
| C          Z-ADD    52953088     LEFTHALFB
| C          MOVE    LEFTHALF      KEYCV
| C*
| C* Calculate the Token Validation value by adding every 4 bytes
| C* and storing the result in the last 4 bytes.
| C*
| C          ADD     KEYTVV1       KEYTVV
| C          ADD     KEYTVV2       KEYTVV
| C          ADD     KEYTVV3       KEYTVV
| C          ADD     KEYTVV4       KEYTVV
| C          ADD     KEYTVV5       KEYTVV
| C          ADD     KEYTVV6       KEYTVV
| C          ADD     KEYTVV7       KEYTVV
| C          ADD     KEYTVV8       KEYTVV
| C          ADD     KEYTVV9       KEYTVV
| C          ADD     KEYTVV10      KEYTVV
| C          ADD     KEYTVV11      KEYTVV
| C          ADD     KEYTVV12      KEYTVV
| C          ADD     KEYTVV13      KEYTVV
| C          ADD     KEYTVV14      KEYTVV
| C          ADD     KEYTVV15      KEYTVV
| C*
| C* Copy token to PINGENKEY
| C*
| C          MOVE    KEYTOKEN      PINGENKEY
| C*
| C*****
| C* Build a PINVER key token
| C*
| C* The control vector for a PINVER key that
| C* has the key part flag set is (in hex):
| C*
| C*      00224200  03480000  00224200  03280000
| C*
| C* If each 4 byte hex part is converted to decimal you get:
| C*
| C*      2260480   55050240  2260480   52953088
| C*
| C* Build the control vector by placing the decimal number in
| C* the appropriate half of the control vector field.
| C          Z-ADD    2245120      LEFTHALFA
| C          Z-ADD    55050240     LEFTHALFB
| C          MOVE    LEFTHALF      KEYCV
| C          Z-ADD    2245120      LEFTHALFA
| C          Z-ADD    52953088     LEFTHALFB

```

```

|      C          MOVE      LEFTHALF      KEYCV
|      C*
|      C* Calculate the Token Validation value by adding every 4 bytes
|      C* and storing the result in the last 4 bytes.
|      C*
|      C          Z-ADD      0          KEYTVV
|      C          ADD       KEYTVV1     KEYTVV
|      C          ADD       KEYTVV2     KEYTVV
|      C          ADD       KEYTVV3     KEYTVV
|      C          ADD       KEYTVV4     KEYTVV
|      C          ADD       KEYTVV5     KEYTVV
|      C          ADD       KEYTVV6     KEYTVV
|      C          ADD       KEYTVV7     KEYTVV
|      C          ADD       KEYTVV8     KEYTVV
|      C          ADD       KEYTVV9     KEYTVV
|      C          ADD       KEYTVV10    KEYTVV
|      C          ADD       KEYTVV11    KEYTVV
|      C          ADD       KEYTVV12    KEYTVV
|      C          ADD       KEYTVV13    KEYTVV
|      C          ADD       KEYTVV14    KEYTVV
|      C          ADD       KEYTVV15    KEYTVV
|      C*
|      C* Copy token to PINVERKEY
|      C*
|      C          MOVE      KEYTOKEN     PINVERKEY
|      C*
|      C*
|      C*****
|      C* Build an IPINENC key token
|      C*
|      C* The control vector for an IPINENC key that
|      C* has the key part flag set is (in hex):
|      C*
|      C*      00215F00 03480000 00215F00 03280000
|      C*
|      C* If each 4 byte hex part is converted to decimal you get:
|      C*
|      C*      2187008 55050240 2187008 52953088
|      C*
|      C*****
|      C* Build the control vector by placing the decimal number in
|      C* the appropriate half of the control vector field.
|      C*****
|      C          Z-ADD      2187008     LEFTHALFA
|      C          Z-ADD      55050240    LEFTHALFB
|      C          MOVE      LEFTHALF     KEYCV
|      C          Z-ADD      2187008     LEFTHALFA
|      C          Z-ADD      52953088    LEFTHALFB
|      C          MOVE      LEFTHALF     KEYCV
|      C*
|      C* Calculate the Token Validation value by adding every 4 bytes
|      C* and storing the result in the last 4 bytes.
|      C*
|      C          Z-ADD      0          KEYTVV
|      C          ADD       KEYTVV1     KEYTVV
|      C          ADD       KEYTVV2     KEYTVV
|      C          ADD       KEYTVV3     KEYTVV
|      C          ADD       KEYTVV4     KEYTVV
|      C          ADD       KEYTVV5     KEYTVV
|      C          ADD       KEYTVV6     KEYTVV
|      C          ADD       KEYTVV7     KEYTVV
|      C          ADD       KEYTVV8     KEYTVV
|      C          ADD       KEYTVV9     KEYTVV
|      C          ADD       KEYTVV10    KEYTVV
|      C          ADD       KEYTVV11    KEYTVV
|      C          ADD       KEYTVV12    KEYTVV
|      C          ADD       KEYTVV13    KEYTVV

```

```

C          ADD      KEYTVV14    KEYTVV
C          ADD      KEYTVV15    KEYTVV
C*
C* Copy token to IPINENC
C*
C          MOVE     KEYTOKEN     IPINKEY
C*
C*
C*****
C* Build an OPINENC key token
C*
C* The control vector for an OPINENC key that
C* has the key part flag set is (in hex):
C*
C*      00247700 03480000 00247700 03280000
C*
C* If each 4 byte hex part is converted to decimal you get:
C*
C*      2389760 55050240 2389760 52953088
C*
C*****
C* Build the control vector by placing the decimal numbers in
C* the appropriate half of the control vector field.
C*****
C          Z-ADD    2389760      LEFTHALFA
C          Z-ADD    55050240     LEFTHALFB
C          MOVE     LEFTHALF     KEYCV
C          Z-ADD    2389760      LEFTHALFA
C          Z-ADD    52953088     LEFTHALFB
C          MOVE     LEFTHALF     KEYCV
C*
C* Calculate the Token Validation value by adding every 4 bytes
C* and storing the result in the last 4 bytes.
C*
C          Z-ADD    0            KEYTVV
C          ADD     KEYTVV1      KEYTVV
C          ADD     KEYTVV2      KEYTVV
C          ADD     KEYTVV3      KEYTVV
C          ADD     KEYTVV4      KEYTVV
C          ADD     KEYTVV5      KEYTVV
C          ADD     KEYTVV6      KEYTVV
C          ADD     KEYTVV7      KEYTVV
C          ADD     KEYTVV8      KEYTVV
C          ADD     KEYTVV9      KEYTVV
C          ADD     KEYTVV10     KEYTVV
C          ADD     KEYTVV11     KEYTVV
C          ADD     KEYTVV12     KEYTVV
C          ADD     KEYTVV13     KEYTVV
C          ADD     KEYTVV14     KEYTVV
C          ADD     KEYTVV15     KEYTVV
C*
C* Copy token to OPINENC
C*
C          MOVE     KEYTOKEN     OPINKEY
C*
C*
C*****
C* Clear key value for PINGEN/PINVER form will be:
C*
C*      01234567 01765432 01234567 01765432
C*
C* The key will be imported into two parts that get excluded
C* OR'ed together. This program uses as key parts:
C*
C*      00224466 00775533 00224466 00775533 and
C*

```

```

C* 01010101 01010101 01010101 01010101
C*
C* Converting these to decimal results in
C*
C* 2245734 7820595 2245734 7820595 and
C*
C* 16843009 16843009 16843009 16843009
C*
C* In this example, the left half of the key is the same as
C* the right half. PIN keys in CCA are double length keys.
C* However, some implementation of DES (including Cryptographic
C* Support/400) use single length keys for PINs. If both
C* halves of a double are the same, then they produce the
C* same output as a single length key, thereby allowing you
C* to exchange data with non-CCA systems.
C*****
C* Import the PINGEN key
C*****
C          MOVEL    'FIRST '    RULEARRAY
C          Z-ADD    1           RULEARRAYCNT
C*****
C* Build the next clear key part by placing the decimal numbers
C* in the appropriate half of the clear key field.
C*****
C          Z-ADD    16843009    LEFTHALFA
C          Z-ADD    16843009    LEFTHALFB
C          MOVEL    LEFTHALF    CLEARKEY
C          MOVE     LEFTHALF    CLEARKEY
C*****
C* Call Key Part Import the first time for the PINGEN key
C*****
C          CALLP    CSNBKPI      (RETURNCODE:
C                               REASONCODE:
C                               EXITDATALEN:
C                               EXITDATA:
C                               RULEARRAYCNT:
C                               RULEARRAY:
C                               CLEARKEY:
C                               PINGENKEY)
C          RETURNCODE  IFGT      4
C          MOVEL    'CSNBKPI'    FAILMESSAGE
C          EXSR     SNDFAILMSG
C          SETON
C          ENDIF
C*****
C* Build the clear key part by placing the decimal number in
C* the appropriate half of the clear key field.
C*****
C          Z-ADD    2245734     LEFTHALFA
C          Z-ADD    7820595     LEFTHALFB
C          MOVEL    LEFTHALF    CLEARKEY
C          MOVE     LEFTHALF    CLEARKEY
C*****
C* Call Key Part Import the second time for the PINGEN key
C*****
C          MOVEL    'LAST '    RULEARRAY
C          CALLP    CSNBKPI      (RETURNCODE:
C                               REASONCODE:
C                               EXITDATALEN:
C                               EXITDATA:
C                               RULEARRAYCNT:
C                               RULEARRAY:
C                               CLEARKEY:
C                               PINGENKEY)
C          RETURNCODE  IFGT      4
C          MOVEL    'CSNBKPI'    FAILMESSAGE
C          EXSR     SNDFAILMSG

```

LR


```

C          SETON          LR
C          ENDIF
C*****
C* Import the PINVER key *
C*****
C          MOVE          'FIRST '    RULEARRAY
C          Z-ADD         1           RULEARRAYCNT
C          Z-ADD         16843009    LEFTHALFA
C          Z-ADD         16843009    LEFTHALFB
C          MOVE          LEFTHALF    CLEARKEY
C          MOVE          LEFTHALF    CLEARKEY
C*****
C* Call Key Part Import the first time for the PINVER key
C*****
C          CALLP         CSNBKPI      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     CLEARKEY:
C                                     PINVERKEY)
C          RETURNCODE   IFGT         4
C          MOVE          'CSNBKPI'    FAILMESSAGE
C          EXSR          SNDFAILMSG
C          SETON
C          ENDIF          LR
C*****
C* Build the clear key part by placing the decimal number in
C* the appropriate half of the clear key field.
C*****
C          Z-ADD         2245734     LEFTHALFA
C          Z-ADD         7820595     LEFTHALFB
C          MOVE          LEFTHALF    CLEARKEY
C          MOVE          LEFTHALF    CLEARKEY
C*****
C* Call Key Part Import the second time for the PINVER key
C*****
C          MOVE          'LAST '     RULEARRAY
C          CALLP         CSNBKPI      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     CLEARKEY:
C                                     PINVERKEY)
C          RETURNCODE   IFGT         4
C          MOVE          'CSNBKPI'    FAILMESSAGE
C          EXSR          SNDFAILMSG
C          SETON
C          ENDIF          LR
C*****
C* Clear key value for IPINENC/OPINENC key pair will be:
C* 012332EF 01020408 012332EF 01020408
C*
C* The key will be imported into two parts that get excluded
C* OR'ed together. This program uses as key parts:
C*
C* 002233EE 00030509 002233EE 00030509 and
C*
C* 01010101 01010101 01010101 01010101
C*
C* Converting these to decimal results in
C*
C* 2241518 197897 2241518 197897 and
C*

```

```

C* 16843009 16843009 16843009 16843009
C*****
C* Import the PINVER key *
C*****
C          MOVEL  'FIRST  '    RULEARRAY
C          Z-ADD  1          RULEARRAYCNT
C*****
C* Build the clear key part by placing the decimal number in
C* the appropriate half of the clear key field.
C*****
C          Z-ADD  16843009    LEFTHALFA
C          Z-ADD  16843009    LEFTHALFB
C          MOVEL  LEFTHALF    CLEARKEY
C          MOVE   LEFTHALF    CLEARKEY
C*****
C* Call Key Part Import the first time for the IPINENC key
C*****
C          CALLP  CSNBKPI      (RETURNCODE:
C                               REASONCODE:
C                               EXITDATALEN:
C                               EXITDATA:
C                               RULEARRAYCNT:
C                               RULEARRAY:
C                               CLEARKEY:
C                               IPINKEY)
C          RETURNCODE  IFGT    4
C          MOVEL      'CSNBKPI'  FAILMESSAGE
C          EXSR      SNDFAILMSG
C          SETON
C          ENDIF
C*****
C* Build the clear key part by placing the decimal number in
C* the appropriate half of the clear key field.
C*****
C          Z-ADD  2241518    LEFTHALFA
C          Z-ADD  197897    LEFTHALFB
C          MOVEL  LEFTHALF    CLEARKEY
C          MOVE   LEFTHALF    CLEARKEY
C*****
C* Call Key Part Import the second time for the IPINENC key
C*****
C          MOVEL  'LAST  '    RULEARRAY
C          CALLP  CSNBKPI      (RETURNCODE:
C                               REASONCODE:
C                               EXITDATALEN:
C                               EXITDATA:
C                               RULEARRAYCNT:
C                               RULEARRAY:
C                               CLEARKEY:
C                               IPINKEY)
C          RETURNCODE  IFGT    4
C          MOVEL      'CSNBKPI'  FAILMESSAGE
C          EXSR      SNDFAILMSG
C          SETON
C          ENDIF
C*****
C* Import the OPINENC key *
C*****
C          MOVEL  'FIRST  '    RULEARRAY
C          Z-ADD  1          RULEARRAYCNT
C*****
C* Build the clear key part by placing the decimal number in
C* the appropriate half of the clear key field.
C*****
C          Z-ADD  16843009    LEFTHALFA
C          Z-ADD  16843009    LEFTHALFB
C          MOVEL  LEFTHALF    CLEARKEY

```

LR

LR

```

C          MOVE      LEFTHALF      CLEARKEY
C*****
C* Call Key Part Import the first time for the OPINENC key
C*****
C          CALLP      CSNBKPI      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     CLEARKEY:
C                                     OPINKEY)
C          RETURNCODE  IFGT      4
C          MOVE      'CSNBKPI'      FAILMESSAGE
C          EXSR      SNDFAILMSG
C          SETON
C          ENDIF
C
C*****
C* Build the clear key part by placing the decimal number in
C* the appropriate half of the clear key field.
C*****
C          Z-ADD      2241518      LEFTHALFA
C          Z-ADD      197897      LEFTHALFB
C          MOVE      LEFTHALF      CLEARKEY
C          MOVE      LEFTHALF      CLEARKEY
C*****
C* Call Key Part Import the second time for the OPINENC key
C*****
C          MOVE      'LAST '      RULEARRAY
C          CALLP      CSNBKPI      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     CLEARKEY:
C                                     OPINKEY)
C          RETURNCODE  IFGT      4
C          MOVE      'CSNBKPI'      FAILMESSAGE
C          EXSR      SNDFAILMSG
C          SETON
C          ENDIF
C
C*****
C* Generate a Clear PIN with CSNBPGN (Clear_PIN_Generate)
C* Rule_array_count = 1
C* Rule_array = "IBM-PIN " (Same as Crypto Support/400)
C* PIN length = 8
C* PIN Check length = 8 (But is ignored for IBM-PIN)
C* Data array:
C* Dec. table set to 0123456789123456
C* validation dta = 1111222233334444
C* clear PIN = ignored
C*****
C          Z-ADD      1          RULEARRAYCNT
C          MOVE      'IBM-PIN '  RULEARRAY
C          Z-ADD      8          PINLEN
C          Z-ADD      8          PINCKL
C          MOVE      '01234567'  DECTABLE
C          MOVE      '89123456'  DECTABLE
C          MOVE      '11112222'  VALDATA
C          MOVE      '33334444'  VALDATA
C*****
C* Call Clear PIN Generate
C*****
C          CALLP      CSNBPGN      (RETURNCODE:
C                                     REASONCODE:

```

LR

LR


```

|      C                                RULEARRAY:
|      C                                PINCKL:
|      C                                DATAARRAY)
|      C      RETURNCODE      IFGT      4
|      C                                MOVEL      'CSNBPVR'      FAILMESSAGE
|      C                                EXSR      SNDFAILMSG
|      C                                SETON
|      C                                ENDIF
|      C*
|      C*****
|      C* Send successful completion message
|      C*****
|      C                                CALL      'QMHSNDPM'
|      C                                PARM      MESSAGEID
|      C                                PARM      MESSAGEFILE
|      C                                PARM      GOODMESSAGE
|      C                                PARM      GOODMSGLLENGTH
|      C                                PARM      MSGTYPE
|      C                                PARM      STACKENTRY
|      C                                PARM      STACKCOUNTER
|      C                                PARM      MSGKEY
|      C                                PARM      ERRCODE
|      C*
|      C                                SETON
|      C*
|      C*****
|      C* Subroutine to send a failure message
|      C*****
|      C      SNDFAILMSG      BEGSR
|      C                                MOVE      FAILMESSAGE      FAILMSGTEXT
|      C                                MOVE      RETURNCODE      FAILRETC
|      C                                MOVE      REASONCODE      FAILRSNC
|      C                                CALL      'QMHSNDPM'
|      C                                PARM      MESSAGEID
|      C                                PARM      MESSAGEFILE
|      C                                PARM      FAILMSG
|      C                                PARM      FAILMSGLLENGTH
|      C                                PARM      MSGTYPE
|      C                                PARM      STACKENTRY
|      C                                PARM      STACKCOUNTER
|      C                                PARM      MSGKEY
|      C                                PARM      ERRCODE
|      C                                ENDSR

```

デジタル署名の生成および検査

デジタル署名と呼ばれる識別値の検査を組み込んでおくことにより、検知されない変更からデータを保護することができます。

デジタル署名の生成

デジタル署名には、ハッシュと公開鍵暗号が使用されます。データに署名すると、データがハッシュされ、ユーザーの秘密鍵でその結果が暗号化されます。暗号化されたハッシュ値をデジタル署名と呼びます。

オリジナルのデータを変更すると、異なるデジタル署名が生成されます。

ファイルに署名するために PKA 鍵を使用するには、プログラムを作成します。

デジタル署名の検証

デジタル署名の検証は、データへの署名と逆の手順を行います。署名を検証すると、署名されたデータが変更されているかが分かります。デジタル署名を検証すると、署名は公開鍵を使用して復号され、

元のハッシュ値が作成されます。署名されたデータはハッシュされています。2つのハッシュ値が一致すると、署名は検証されたこととなります。検証するには、プログラムを作成します。

法律に関する重要な情報については、292ページの『コードに関するライセンス情報および特記事項』を参照してください。

関連概念

149ページの『DES キーおよび PKA キーの作成』
DES および PKA 鍵を作成して、DES 鍵ストアに保管することができます。

例: 暗号化コプロセッサによるファイルの署名:

暗号化コプロセッサでファイルに署名するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292ページの『コードに関するライセンス情報および特記事項』を参照してください。

付属のプログラム例を使用する場合には、必要に応じてそのプログラムを変更してください。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
/*-----*/
/* Description:  Digitally signs a streams file.          */
/*                                                     */
/* COPYRIGHT      5769-SS1 (c) IBM Corp 1999          */
/*                                                     */
/* This material contains programming source code for your
/* consideration.  These examples have not been thoroughly
/* tested under all conditions.  IBM, therefore, cannot
/* guarantee or imply reliability, serviceability, or function
/* of these programs.  All programs contained herein are
/* provided to you "AS IS".  THE IMPLIED WARRANTIES OF
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
/* EXPRESSLY DISCLAIMED.  IBM provides no program services for
/* these programs and files.
/*
/* Parameters:  File to be signed
/*              File to contain signature
/*              Key label of key to use
/*
/* Examples:
/* CALL PGM(SIGNFILE) PARM('file_to_sign' 'file_to_hold_sign'
/*                          'key_label');
/*
/* Note: The CCA verbs used in the this program are more fully
/* described in the IBM CCA Basic Services Reference
/* and Guide (SC31-8609) publication.
/*
/* Note: This program assumes the card you want to use is
/* already identified either by defaulting to the CRP01
/* device or has been explicitly named using the
/* Cryptographic_Resource_Allocate verb.  Also this
/* device must be varied on and you must be authorized
/* to use this device description.
/*
/* Use the following commands to compile this program:
/* ADDLIB LIB(QCCA)
/* CRTCMOD MODULE(SIGNFILE) SRCFILE(SAMPLE) SYSIFCOPT(*IFSIO)
/* CRTPGM PGM(SIGNFILE) MODULE(SIGNFILE)
/*          BNDSRVPGM(QCCA/CSNDDSG QCCA/CSNBOWH)
/*
/* Note: authority to the CSNDDSG and CSNBOWH service programs
/* in the QCCA library is assumed.
```

```

| /*                                                     */
| /* Common Cryptographic Architecture (CCA) verbs used: */
| /*   Digital_Signature_Generate (CSNDDSG)             */
| /*   One_Way_Hash (CSNBOWH)                          */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h" /* header file for CCA Cryptographic
|                       Service Provider */
|
| /*-----*/
| /* standard return codes */
| /*-----*/
| #define ERROR -1
| #define OK 0
|
| int hash_file(long h_len, char h_out[128], FILE *t_in);
|
| int main(int argc, char *argv[])
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|     long return_code;
|     long reason_code;
|     long exit_data_length = 0L;
|     char exit_data[2];
|     long rule_array_count = 0L;
|     char rule_array[1][8];
|
|     /*-----*/
|     /* parameters unique to this sample program */
|     /*-----*/
|     long PKA_private_key_identifier_length = 64;
|     char PKA_private_key_identifier[64];
|     long hash_length = 16L;
|     char hash[128];
|     long signature_field_length = 128L;
|     long signature_bit_length = 0L;
|     char signature_field[256];
|     char key_label[64];
|     long key_token_length = 2500L;
|     char key_token[2500];
|
|     FILE *file2sign;
|     FILE *signature;
|     int hash_return;
|
|     if (argc < 2)
|     {
| printf("Name of file to be signed is missing.");
| return ERROR;
|     }
|     else if (argc < 3)
|     {
| printf("Name of file where the signature should ");
| printf("be written is missing.");
| return ERROR;
|     }
|     else if (argc < 4)
|     {
| printf("Key label for the key to be used for signing is missing.");
| return ERROR;
|     }
| }

```

```

|         if ( (strlen(argv[3])) > 64 )
|         {
| printf("Invalid Key Label. Key label longer than 64.");
| return ERROR;
|         }
|         else
|         {
| memset(PKA_private_key_identifier, ' ', 64);
| memcpy(PKA_private_key_identifier, argv[3],strlen(argv[3]));
|         }
|
|         /* Open the file that is being signed. */
|         if ( (file2sign = fopen(argv[1],"rb")) == NULL)
|         {
| printf("Opening of file %s failed.",argv[1]);
| return ERROR;
|         }
|
|         /* Obtain a hash value for the file. */
|         hash_return = hash_file(hash_length, hash, file2sign);
|
|         /* Close the file. */
|         fclose(file2sign);
|
|         if (hash_return != OK)
|         {
| printf("Signature generation failed due to hash error.¥n");
|         }
|
|         else
|         {
| /* Use CSNDDSG to generate the signature. */
| CSNDDSG(&return_code,
| &reason_code,
| &exit_data_length,
| exit_data,
| &rule_array_count,
| (char *) rule_array,
| &PKA_private_key_identifier_length,
| PKA_private_key_identifier,
| &hash_length,
| hash,
| &signature_field_length,
| &signature_bit_length,
| signature_field);
|         }
|
|         if (return_code != 0)
|         {
| printf("Signature generation failed with return/reason code %ld/%ld",
| return_code, reason_code);
| return ERROR;
|         }
|         else
|         {
| printf("Signature generation was successful.");
| printf("Return/Reason codes = %ld/%ld¥n", return_code, reason_code);
| printf("Signature has length = %ld¥n",signature_field_length);
|
|         signature = fopen(argv[2],"wb");
|         if (signature == NULL)
|         {
| printf("Open of file %s failed.",argv[2]);
| printf("Signature was not saved.");

```



```

|   return ERROR;
|   }
|
|   fwrite(signature_field, 1, signature_field_length, signature);
|   fclose(signature);
|   printf("Signature was saved successfully in %s.", argv[2]);
|   return OK;
|   }
| }
|
| int hash_file(long h_len, char h_out[128], FILE *t_in)
| {
|   /*-----*/
|   /* standard CCA parameters */
|   /*-----*/
|   long return_code;
|   long reason_code;
|   long exit_data_length = 0;
|   char exit_data[2];
|   long rule_array_count = 2;
|   char rule_array[2][8];
|
|   /*-----*/
|   /* parameters unique to this function */
|   /*-----*/
|   long text_length;
|   char text[1024];
|   long chaining_vector_length = 128;
|   char chaining_vector[128];
|
|   long file_length;
|
|   fseek(t_in, 0, SEEK_END);
|   file_length = ftell(t_in);
|   rewind(t_in);
|
|   text_length = fread(text, 1, 1024, t_in);
|
|   memcpy(rule_array[0], "MD5    ", 8);
|
|   if (file_length <= 1024) {
|   memcpy(rule_array[1], "ONLY   ", 8);
|   }
|   else {
|   memcpy(rule_array[1], "FIRST  ", 8);
|   }
|
|   while (file_length > 0)
|   {
|   CSNBOWH(&return_code,
|   &reason_code,
|   &exit_data_length,
|   exit_data,
|   &rule_array_count,
|   (char *) rule_array,
|   &text_length,
|   text,
|   &chaining_vector_length,
|   chaining_vector,
|   &h_len,
|   h_out);
|
|   if (return_code != 0)
|       break;
|
|   printf("Hash iteration worked.¥n");

```

```

| file_length -= text_length;
|
| if (file_length > 0)
| {
|     text_length = fread(text, 1, 1024, t_in);
|
|     if (file_length <= 1024) {
| memcpy(rule_array[1], "LAST   ", 8);
|     }
|     else {
| memcpy(rule_array[1], "MIDDLE ", 8);
|     }
| }
|
| if (return_code != 0)
| {
| printf("Hash function failed with return/reason code %ld/%ld\n",
|     return_code, reason_code);
| return ERROR;
| }
| else
| {
| printf("Hash completed successfully.\n");
| printf("hash length = %ld\n", h_len);
| printf("hash = %.32s\n", h_out);
| return OK;
| }
| }

```

| 例: 暗号化コプロセッサによるデジタル署名の検証:

| 暗号化コプロセッサでデジタル署名を検証するには、必要に応じて以下のプログラム例を変更してください。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| /*-----*/
| /* Description: Verifies the digital signature of an IFS file */
| /*               produced by the SIGNFILE sample program. */
| /*               */
| /* COPYRIGHT     5769-SS1 (c) IBM Corp 1999 */
| /*               */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these programs. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /*               */
| /* Parameters: Signed file */
| /*               File containing the signature */
| /*               Key label of the key to use */
| /*               */
| /* Examples: */
| /* CALL PGM(VERFILESIG) PARM('name_of_signed_file' + */
| /*                             'name_of_file_w_signature' + */
| /*                             'key_label'); */
| /* Note: The CCA verbs used in the this program are more fully */
| /* described in the IBM CCA Basic Services Reference */
| /* and Guide (SC31-8609) publication. */

```

```

| /* */
| /* Note: This program assumes the card you want to use is */
| /* already identified either by defaulting to the CRP01 */
| /* device or has been explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* Use the following commands to compile this program: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(VERFILESIG) SRCFILE(SAMPLE) SYSIFCOPT(*IFSIO)*/
| /* CRTPGM PGM(SIGNFILE) MODULE(SIGNFILE) + */
| /* BNSRVPGM(QCCA/CSNDDSV QCCA/CSNBOWH) */
| /* */
| /* Note: authority to the CSNDDSV and CSNBOWH service programs */
| /* in the QCCA library is assumed. */
| /* */
| /* Common Cryptographic Architecture (CCA) verbs used: */
| /* Digital_Signature_Verify (CSNDDSV) */
| /* One_Way_Hash (CSNBOWH) */
| /*-----*/
|
| #include <stdlib.h>
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h" /* header file for CCA Cryptographic
| Service Provider */
|
| /*-----*/
| /* standard return codes */
| /*-----*/
| #define ERROR -1
| #define OK 0
|
| int hash_file(long h_len, char h_out[128], FILE *t_in);
|
| int main(int argc, char *argv[])
| {
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code;
| long reason_code;
| long exit_data_length = 0L;
| char exit_data[2];
| long rule_array_count = 0L;
| char rule_array[1][8];
|
| /*-----*/
| /* parameters unique to this sample program */
| /*-----*/
| long PKA_public_key_identifier_length = 64;
| char PKA_public_key_identifier[64];
| long hash_length = 16L;
| char hash[128];
| long signature_field_length;
| char signature_field[256];
| char key_label[64];
|
| FILE *file2verify;
| FILE *signature;
| int hash_return;
|
| if (argc < 2)
| {
| printf("Name of file to be verified is missing.\%n");

```

```

| return ERROR;
|     }
|     else if (argc < 3)
|     {
| printf("Name of file containing the signature is missing.¥n");
| return ERROR;
|     }
|     else if (argc < 4)
|     {
| printf("Key label for the key to be used for verification is missing.¥n");
| return ERROR;
|     }

|
|     if (strlen(argv[3]) > 64 )
|     {
| printf("Invalid Key Label. Key label longer than 64 bytes.");
| return ERROR;
|     }
|     else
|     {
| memset(PKA_public_key_identifier, ' ', 64);
| memcpy(PKA_public_key_identifier, argv[3], strlen(argv[3]));
|     }

|     /* Open the file that is being verified. */
|     if ( (file2verify = fopen(argv[1],"rb")) == NULL)
|     {
| printf("Opening of file %s failed.",argv[1]);
| return ERROR;
|     }

|     /* Obtain a hash value for the file. */
|     hash_return = hash_file(hash_length, hash, file2verify);

|     /* Close the file. */
|     fclose(file2verify);

|     if (hash_return != OK)
|     {
| printf("Signature verification failed due to hash error.¥n");
| return ERROR;
|     }
|     else
|     {
| signature = fopen(argv[2],"rb");
| if (signature == NULL)
| {
|     printf("Open of signature file %s failed.",argv[2]);
|     printf("Signature was not verified.");
|     return ERROR;
| }
| }

| memset(signature_field, ' ', 256);

| fseek(signature, 0, SEEK_END);
| signature_field_length = ftell(signature);
| rewind(signature);

| fread(signature_field, 1, signature_field_length, signature);
| fclose(signature);

| /* Use CSNDDSV to verify the signature. */
| CSNDDSV(&return_code,
| &reason_code,
| &exit_data_length,
| exit_data,

```

```

|     &rule_array_count,
|     (char *) rule_array,
|     &PKA_public_key_identifier_length,
|     PKA_public_key_identifier,
|     &hash_length,
|     hash,
|     &signature_field_length,
|     signature_field);
|     }
|
|     if (return_code != 0)
|     {
| printf("Signature verification failed with return/reason code %ld/%ld",
| return_code, reason_code);
| return ERROR;
|     }
|     else
|     {
| printf("Signature verification was successful.");
| printf("Return/Reason codes = %ld/%ld\n", return_code, reason_code);
|
|     }
| }
|
| int hash_file(long h_len, char h_out[128], FILE *t_in)
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|
|     long return_code;
|     long reason_code;
|     long exit_data_length = 0;
|     char exit_data[2];
|     long rule_array_count = 2;
|     char rule_array[2][8];
|
|     /*-----*/
|     /* parameters unique to this function */
|     /*-----*/
|     long text_length;
|     char text[1024];
|     long chaining_vector_length = 128;
|     char chaining_vector[128];
|
|     long file_length;
|
|     fseek(t_in, 0, SEEK_END);
|     file_length = ftell(t_in);
|     rewind(t_in);
|
|     text_length = fread(text, 1, 1024, t_in);
|
|     memcpy(rule_array[0], "MD5      ", 8);
|
|     if (file_length <= 1024) {
| memcpy(rule_array[1], "ONLY   ", 8);
|     }
|     else {
| memcpy(rule_array[1], "FIRST  ", 8);
|     }
|
|     while (file_length > 0)
|     {

```

```

| CSNBOWH(&return_code,
|   &reason_code,
|   &exit_data_length,
|   exit_data,
|   &rule_array_count,
|   (char *) rule_array,
|   &text_length,
|   text,
|   &chaining_vector_length,
|   chaining_vector,
|   &h_len,
|   h_out);
|
| if (return_code != 0)
|     break;
|
| printf("Hash iteration worked.¥n");
|
| file_length -= text_length;
|
| if (file_length > 0)
| {
|     text_length = fread(text, 1, 1024, t_in);
|
|     if (file_length <= 1024) {
| memcpy(rule_array[1], "LAST   ", 8);
|     }
|     else {
| memcpy(rule_array[1], "MIDDLE ", 8);
|     }
| }
|
| if (return_code != 0)
| {
| printf("Hash function failed with return/reason code %ld/%ld¥n",
|     return_code, reason_code);
| return ERROR;
| }
| else
| {
| printf("Hash completed successfully.¥n");
| printf("hash length = %ld¥n", h_len);
| printf("hash = %.32s¥n¥n", h_out);
| return OK;
| }
| }

```

複数の暗号化コプロセッサの管理

区画ごとに最大 8 つの暗号化コプロセッサを持つことができます。サーバーごとにサポートされる暗号化コプロセッサの最大数は、システム・モードによって異なります。複数のコプロセッサを SSL で使用する場合は、このトピックを参照してください。

複数の暗号化コプロセッサおよび複数のジョブに作業を振り分けると、それらがすべて同じ構成であれば、パフォーマンスが向上します。1 つのジョブには、一度に 1 つのコプロセッサ (暗号装置記述) しか割り振ることができません。ただし、現行のコプロセッサの割り振りを解除して、新しいコプロセッサを割り振ることにより、コプロセッサ間でジョブを切り替えることができます。i5/OS SSL ユーザーの場合は、DCM の SSL 構成で SSL セッションの確立に複数のコプロセッサを使用するように指定されていれば、コプロセッサの割り振りと割り振り解除は、システムによって管理されます。

コプロセッサをすべて同じに構成すると、すべての操作鍵はすべてのコプロセッサで同じように作用します。あるコプロセッサで暗号化されたデータを、別のコプロセッサで暗号化解除することができます。

す。すべての鍵ストア・ファイルは、コプロセッサを交換しても動作します。コプロセッサを同じように構成する場合の最も重要なパーツはマスター鍵です。あるコプロセッサに対してマスター鍵をパーツに入力した場合に、そのマスター鍵のパーツを他のコプロセッサでも区別なく動作するようにするには、他のすべてのコプロセッサに対しても、同じマスター鍵のパーツを入力しなければなりません。コプロセッサ内でランダムなマスター鍵が生成された場合、すべてのコプロセッサが区別なく動作するようにするには、他のコプロセッサにもそのマスター鍵を複製しなければなりません。

すべてのコプロセッサを同じ構成にしたい場合もあります。コプロセッサの構成をすべて異なるものにしたり、コプロセッサを複数のグループでセットアップし、グループ内では構成を同じにし、異なるグループ間では構成を異なるようにすることもできます。このような場合、すべての操作鍵がすべてのコプロセッサで同じようには働かない可能性があります。あるコプロセッサで暗号化されたデータが、別のコプロセッサで回復できないこともあります。また、コプロセッサが異なると鍵ストア・ファイルが動作しないこともあります。このような場合は、どの鍵ストア・ファイルと操作鍵が、特定のコプロセッサに対して動作するのかを追跡しなければなりません。コプロセッサを異なる構成にすると、暗号アプリケーションのスケラビリティが制限されることがありますが、セキュリティの面では、より高い細分度が提供されます。例えば、異なる暗号装置記述には異なるオブジェクト権限を認可することができます。

保管 PKA 鍵を使用している場合は、コプロセッサも交換可能にはなりません。保管鍵は、いかなる場合でもコプロセッサの外部にエクスポートすることはできません。したがって、その保管鍵を使用する暗号要求は、その保管鍵を保管しているコプロセッサに送信しなければなりません。

以下の解説は、i5/OS アプリケーションを使用している場合にのみ適用されます。

関連概念

98 ページの『DCM および SSL で使用するための暗号化コプロセッサの構成』
暗号化コプロセッサを SSL で使用できるようにする場合は、ここに記載された情報を参照してください。

装置の割り振り: `Cryptographic_Resource_Allocate (CSUACRA) API verb` は、すべての連続した暗号要求を経路指定する方法をシステムが判別できるように、暗号装置をジョブに対し明示的に割り振る場合に使用されます。最初に `Cryptographic_Resource_Allocate (CSUACRA) API verb` を明示的に使用せずに `CCA API` を使用すると、システムはデフォルトの暗号装置を割り振ります。デフォルトの装置は、`CRP01` という名前の暗号装置です。この装置は、基本構成ウィザードまたは、装置記述の作成 (暗号) (`CRTDEVCRP`) `CL` コマンドのいずれかを使用して作成しなければなりません。デフォルトの暗号装置以外の装置を使用したい場合は、`CSUACRA` のみを使用します。ジョブに割り振られた装置は、割り振りが明示的であっても暗黙的であっても、ジョブが終了するか、`Cryptographic_Resource_Deallocate (CSUACRD) API verb` を使用してその装置の割り振りを解除するまで、割り振られた状態にあります。

関連資料

186 ページの『例: コプロセッサを割り振るための ILE C プログラム』
コプロセッサを割り振るには、必要に応じて以下のプログラム例を変更してください。

188 ページの『例: コプロセッサを割り振るための ILE RPG プログラム』
コプロセッサを割り振るには、必要に応じて以下のプログラム例を変更してください。

装置の割り振り解除: 暗号化コプロセッサを使い終わった後で、`Cryptographic_Resource_Deallocate (CSUACRD) API verb` を使用して、暗号化コプロセッサの割り振りを解除してください。暗号装置記述は、その装置を使用しているジョブがすべてその装置の割り振りを解除するまでオフに変更することはできません。

関連資料

| 190 ページの『例: コプロセッサの割り振りを解除するための ILE C プログラム』
 | コプロセッサの割り振りを解除するには、必要に応じて以下のプログラム例を変更してください。
 | 192 ページの『例: コプロセッサの割り振りを解除するための ILE RPG プログラム』
 | コプロセッサの割り振りを解除するには、必要に応じて以下のプログラム例を変更してください。

| **例: コプロセッサを割り振るための ILE C プログラム:**

| コプロセッサを割り振るには、必要に応じて以下のプログラム例を変更してください。

| **注:** 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
 | を参照してください。

```
| /*-----*/
| /* Allocate a crypto device to the job. */
| /* */
| /* */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000 */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* none. */
| /* */
| /* Example: */
| /* CALL PGM(CRPALLOC) (CRP02) */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Cryptographic_Resource_Allocate (CSUACRA). */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(CRPALLOC) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(CRPALLOC) MODULE(CRPALLOC) */
| /* BNDSRVPGM(QCCA/CSUACRA) */
| /* */
| /* Note: Authority to the CSUACRA service program in the */
| /* QCCA library is assumed. */
| /* */
| /*-----*/
| #include <string.h>
| #include <stdio.h>
| #include "csucincl.h"
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
| #define WARNING 4
```



```

| int main(int argc, char *argv[])
| {
|     /*-----*/
|     /* standard CCA parameters */
|     /*-----*/
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[2][8];
|     long rule_array_count = 2;
|     long resource_name_length;
|
|     /*-----*/
|     /* Process the parameters */
|     /*-----*/
|     if (argc < 1)
|     {
|         printf("Device parameter must be specified.¥n");
|         return(ERROR);
|     }
|
|     /*-----*/
|     /* Set the keyword in the rule array */
|     /*-----*/
|     memcpy(rule_array,"DEVICE ",8);
|     rule_array_count = 1;
|
|     /*-----*/
|     /* Set the resource name length */
|     /*-----*/
|     resource_name_length = strlen(argv[1]);
|
|     /*-----*/
|     /* Call Cryptographic Resource Allocate SAPI */
|     /*-----*/
|     CSUACRA( &return_code, &reason_code, &exit_data_length,
|             (char *)exit_data,
|             (long *) &rule_array_count,
|             (char *) rule_array,
|             (long *) &resource_name_length,
|             (char *) argv[1]); /* resource name */
|
|     /*-----*/
|     /* Check the return code and display the results */
|     /*-----*/
|     if ( (return_code == OK) | (return_code == WARNING) )
|     {
|         printf("Request was successful¥n");
|         return(OK);
|     }
|     else
|     {
|         printf("Request failed with return/reason codes: %d/%d ¥n",
|             return_code, reason_code);
|         return(ERROR);
|     }
| }

```

関連概念

185 ページの『装置の割り振り』

例: コプロセッサを割り振るための ILE RPG プログラム:

コプロセッサを割り振るには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```
D*****
D* CRPALLOC
D*
D* Sample program that allocates a crypto device to the job.
D*
D*
D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
D*
D* This material contains programming source code for your
D* consideration. These example has not been thoroughly
D* tested under all conditions. IBM, therefore, cannot
D* guarantee or imply reliability, serviceability, or function
D* of these programs. All programs contained herein are
D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
D* these programs and files.
D*
D*
D* Note: Input format is more fully described in Chapter 2 of
D* IBM CCA Basic Services Reference and Guide
D* (SC31-8609) publication.
D*
D* Parameters:
D* Device Name
D*
D* Example:
D* CALL PGM(CRPALLOC) PARM(CRP02)
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(CRPALLOC) SRCFILE(SAMPLE)
D* CRTPGM PGM(CRPALLOC) MODULE(CRPALLOC)
D* BNDSRVPGM(QCCA/CSUACRA)
D*
D* Note: Authority to the CSUACRA service program in the
D* QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Cryptographic_Resource_Allocate (CSUACRA)
D*
D*-----
D* Declare variables for CCA SAPI calls
D*-----
D* ** Return code
DRETURNCODE S 9B 0
D* ** Reason code
DREASONCODE S 9B 0
D* ** Exit data length
DEXITDATALEN S 9B 0
D* ** Exit data
DEXITDATA S 4
D* ** Rule array count
DRULEARRAYCNT S 9B 0
D* ** Rule array
DRULEARRAY S 16
D* ** Resource name length
DRESOURCENAMLEN S 9B 0
D* ** Resource name
DRESOURCENAME S 10
```

```

D*
D*****
D* Prototype for Cryptographic_Resource_Allocate (CSUACRA)
D*****
DCSUACRA          PR
DRETCODE          9B 0
DRSNCODE          9B 0
DEXTDTALEN       9B 0
DEXTDTA           4
DRARRAYCT        9B 0
DRARRAY           16
DRSCNAMLEN       9B 0
DRSCNAM           10
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG             S          75    DIM(2) CTDATA PERRCD(1)
DMSGLENGTH       S          9B 0  INZ(75)
D                DS
DMSGTEXT         1          75
DFAILRETC        41        44
DFAILRSNC        46        49
DMESSAGEID       S          7    INZ('      ')
DMESSAGEFILE     S          21    INZ('          ')
DMSGKEY          S          4     INZ('      ')
DMSGTYPE         S          10    INZ('*INFO ')
DSTACKENTRY      S          10    INZ('*      ')
DSTACKCOUNTER    S          9B 0  INZ(2)
DERRCODE         DS
DBYTESIN         1          4B 0  INZ(0)
DBYTESOUT        5          8B 0  INZ(0)
D*
C*****
C* START OF PROGRAM *
C* *
C*-----*
C  *ENTRY          PLIST
C                  PARM                RESOURCENAME  10
C* *
C*-----*
C* Set the keyword in the rule array *
C*-----*
C          MOVEL   'DEVICE '  RULEARRAY
C          Z-ADD   1          RULEARRAYCNT
C*
C*-----*
C* Set the resource name length *
C*-----*
C          Z-ADD   10          RESOURCENAMLEN
C*
C*-----*
C* Call Cryptographic Resource Allocate SAPI *
C*-----*
C          CALLP   CSUACRA    (RETURNCODE:
C                          REASONCODE:
C                          EXITDTALEN:
C                          EXITDATA:
C                          RULEARRAYCNT:
C                          RULEARRAY:
C                          RESOURCENAMLEN:
C                          RESOURCENAME)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      4

```

```

| C*          *-----*
| C*          * Send error message *
| C*          *-----*
| C              MOVE      MSG(1)      MSGTEXT
| C              MOVE      RETURNCODE  FAILRETC
| C              MOVE      REASONCODE  FAILRSNC
| C              EXSR      SNDMSG
| C*
| C              ELSE
| C*
| C*          *-----*
| C*          * Send success message *
| C*          *-----*
| C              MOVE      MSG(2)      MSGTEXT
| C              EXSR      SNDMSG
| C*
| C              ENDIF
| C*
| C              SETON                      LR
| C*
| C*****
| C* Subroutine to send a message
| C*****
| C      SNDMSG      BEGSR
| C              CALL      'QMHSNDPM'
| C              PARM      MESSAGEID
| C              PARM      MESSAGEFILE
| C              PARM      MSGTEXT
| C              PARM      MSGLENGTH
| C              PARM      MSGTYPE
| C              PARM      STACKENTRY
| C              PARM      STACKCOUNTER
| C              PARM      MSGKEY
| C              PARM      ERRCODE
| C              ENDSR
| C*

```

```

| **
| CSUACRA failed with return/reason codes 9999/9999'
| The request completed successfully

```

関連概念

185 ページの『装置の割り振り』

例: コプロセッサの割り振りを解除するための ILE C プログラム:

コプロセッサの割り振りを解除するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| /*-----*/
| /* Deallocate a crypto device from a job. */
| /* */
| /* */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000 */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */

```

```

| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* none. */
| /* */
| /* Example: */
| /* CALL PGM(CRPDEALLOC) (CRP02) */
| /* */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Cryptographic_Resource_Deallocate (CSUACRD). */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(CRPALLOC) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(CRPALLOC) MODULE(CRPALLOC) */
| /* BNDSRVPGM(QCCA/CSUACRD) */
| /* */
| /* Note: Authority to the CSUACRD service program in the */
| /* QCCA library is assumed. */
| /* */
| /*-----*/
| #include <string.h>
| #include <stdio.h>
| #include "csucincl.h"
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
| #define WARNING 4
|
| int main(int argc, char *argv[])
|
| {
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
| long return_code = 0;
| long reason_code = 0;
| long exit_data_length = 2;
| char exit_data[4];
| char rule_array[2][8];
| long rule_array_count = 2;
| long resource_name_length;
|
| /*-----*/
| /* Process the parameters */
| /*-----*/
| if (argc < 1)
| {
| printf("Device parameter must be specified.\n");
| return(ERROR);
| }
|
| /*-----*/
| /* Set the keyword in the rule array */
| /*-----*/
| memcpy(rule_array,"DEVICE ",8);
| rule_array_count = 1;
|
| /*-----*/

```

```

| /* Set the resource name length */
| /*-----*/
| resource_name_length = strlen(argv[1]);
|
| /*-----*/
| /* Call Cryptographic Resource Deallocate SAPI */
| /*-----*/
| CSUACRD( &return_code, &reason_code, &exit_data_length,
|         (char *)exit_data,
|         (long *) &rule_array_count,
|         (char *) rule_array,
|         (long *) &resource_name_length,
|         (char *) argv[1]); /* resource name */
|
| /*-----*/
| /* Check the return code and display the results */
| /*-----*/
| if ( (return_code == OK) | (return_code == WARNING) )
| {
|     printf("Request was successful\n");
|     return(OK);
| }
| else
| {
|     printf("Request failed with return/reason codes: %d/%d \n",
|           return_code, reason_code);
|     return(ERROR);
| }
| }

```

関連概念

185 ページの『装置の割り振り解除』

例: コプロセッサの割り振りを解除するための ILE RPG プログラム:

コプロセッサの割り振りを解除するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* CRPDEALLOC
| D*
| D* Sample program that deallocates a crypto device to the job.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D* IBM CCA Basic Services Reference and Guide
| D* (SC31-8609) publication.
| D*
| D* Parameters:
| D* Device name
| D*

```

```

| D* Example:
| D* CALL PGM(CRPDEALLOC) PARM(CRP02)
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(CRPDEALLOC) SRCFILE(SAMPLE)
| D* CRTPGM PGM(CRPDEALLOC) MODULE(CRPDEALLOC)
| D* BNSRVPGM(QCCA/CSUACRD)
| D*
| D* Note: Authority to the CSUACRD service program in the
| D* QCCA library is assumed.
| D*
| D* The Common Cryptographic Architecture (CCA) verbs used are
| D* Cryptographic_Resource_Deallocate (CSUACRD)
| D*
| D*-----
| D* Declare variables for CCA SAPI calls
| D*-----
| D* ** Return code
| DRETURNCODE S 9B 0
| D* ** Reason code
| DREASONCODE S 9B 0
| D* ** Exit data length
| DEXITDATALEN S 9B 0
| D* ** Exit data
| DEXITDATA S 4
| D* ** Rule array count
| DRULEARRAYCNT S 9B 0
| D* ** Rule array
| DRULEARRAY S 16
| D* ** Resource name length
| DRESOURCENAMLEN S 9B 0
| D* ** Resource name
| DRESOURCENAME S 10
| D*
| D*****
| D* Prototype for Cryptographic_Resource_Deallocate (CSUACRD)
| D*****
| DCSUACRD PR
| DRETCODE 9B 0
| DRSNCODE 9B 0
| DEXTDTALEN 9B 0
| DEXTDTA 4
| DRARRAYCT 9B 0
| DRARRAY 16
| DRSCNAMLEN 9B 0
| DRSCNAM 10
| D*
| D*-----
| D* ** Declares for sending messages to the
| D* ** job log using the QMHSNDPM API
| D*-----
| DMSG S 75 DIM(2) CTDATA PERRCD(1)
| DMSGLENGTH S 9B 0 INZ(75)
| D DS
| DMSGTEXT 1 75
| DFAILRETC 41 44
| DFAILRSNC 46 49
| DMESSAGEID S 7 INZ(' ')
| DMESSAGEFILE S 21 INZ(' ')
| DMSGKEY S 4 INZ(' ')
| DMSGTYPE S 10 INZ('*INFO ')
| DSTACKENTRY S 10 INZ('* ')
| DSTACKCOUNTER S 9B 0 INZ(2)
| DERRCODE DS
| DBYTESIN 1 4B 0 INZ(0)
| DBYTESOUT 5 8B 0 INZ(0)

```

```

D*
C*****
C* START OF PROGRAM *
C* *
C*-----*
C   *ENTRY      PLIST
C               PARM                RESOURCENAME
C*-----*
C* Set the keyword in the rule array *
C*-----*
C               MOVE      'DEVICE '  RULEARRAY
C               Z-ADD     1          RULEARRAYCNT
C*-----*
C* Set the resource name length *
C*-----*
C               Z-ADD     10         RESOURCENAMLEN
C*-----*
C* Call Cryptographic Resource Deallocate SAPI *
C*-----*
C               CALLP     CSUACRD    (RETURNCODE:
C                                   REASONCODE:
C                                   EXITDATALEN:
C                                   EXITDATA:
C                                   RULEARRAYCNT:
C                                   RULEARRAY:
C                                   RESOURCENAMLEN:
C                                   RESOURCENAME)
C*-----*
C* Check the return code *
C*-----*
C   RETURNCODE  IFGT      4
C*-----*
C* * Send error message *
C*-----*
C               MOVE      MSG(1)     MSGTEXT
C               MOVE      RETURNCODE  FAILRETC
C               MOVE      REASONCODE  FAILRSNC
C               EXSR      SNDMSG
C*-----*
C               ELSE
C*-----*
C* * Send success message *
C*-----*
C               MOVE      MSG(2)     MSGTEXT
C               EXSR      SNDMSG
C*-----*
C               ENDIF
C*-----*
C               SETON
C*-----*
C*****
C* Subroutine to send a message
C*****
C   SNDMSG      BEGSR
C               CALL      'QMHSNDPM'
C               PARM      MESSAGEID
C               PARM      MESSAGEFILE
C               PARM      MSGTEXT
C               PARM      MSGLENGTH
C               PARM      MSGTYPE
C               PARM      STACKENTRY
C               PARM      STACKCOUNTER
C               PARM      MSGKEY
C               PARM      ERRCODE

```

LR


```
| C ENDSR
| C*
| **
| CSUACRD failed with return/reason codes 9999/9999'
| The request completed successfully
```

関連概念

185 ページの『装置の割り振り解除』

マスター鍵の複製

マスター鍵の複製は、ある暗号化コプロセッサから別のコプロセッサに、マスター鍵の値を露出しない
で安全にコピーする方法です。複数のコプロセッサを SSL で使用する場合は、このトピックを参照して
ください。

これを実行するには、マスター鍵を n 個の共用パーツに分割します。 n は 1 から 15 の数です。別のコ
プロセッサでマスター鍵を再作成するには、 m 個の共用パーツが必要になります。ここで、 m は 1 から
15 の範囲の、 n 以下の数です。

「複製」という用語は、そのプロセスを「コピー」と区別するために使用されます。1 つの共用パーツ、ま
たは m 個よりも少ない共用パーツの組み合わせでは、マスター鍵を再作成するために必要な情報が十分に
得られないためです。

複製するマスター鍵を含むコプロセッサは、マスター鍵共用ソース・ノード、あるいは送信側と呼ばれま
す。送信側は、保管 RSA 鍵のペアを生成しなければなりません。この秘密鍵は、その生成時に、複製での
使用に適している、とマークされていなければなりません。この秘密鍵は、コプロセッサ共用署名鍵ある
いは送信側鍵として知られています。マスター鍵を受け取るコプロセッサは、マスター鍵共用ターゲッ
ト・ノード、あるいは受信側と呼ばれます。受信側も保管 RSA 鍵のペアを生成し、複製での使用に適して
いる、とマークされていなければなりません。この鍵は、コプロセッサ共用受信鍵あるいは単に受信側鍵
として知られています。

送信側および受信側の公開鍵は共に、コプロセッサ内の保管秘密鍵でデジタル署名されているか、ディ
ジタル認証されていなければなりません。このコプロセッサ内の保管秘密鍵を、公開鍵認証ノード、ある
いは認証者と呼びます。この保管秘密鍵は認証者鍵です。また、共用管理鍵とも呼ばれます。関連付けら
れている公開鍵を、送信側および受信側の両方に登録してはじめて、共用パーツを生成および登録する
ことができます。暗号化コプロセッサは、認証者のみの役割を担うか、送信側と受信側の両方になるか、
認証者と受信側の両方になることができます。

各共用パーツが生成されると、コプロセッサが送信側の秘密鍵を使ってそれに署名し、新規に作成された
Triple-DES 鍵を使って暗号化します。次に triple-DES 鍵が受信側の公開鍵によりラップまたは暗号化され
ます。

各共用パーツが受信されると、その共用パーツの署名が送信側の公開鍵を使用して検査され、 Triple-DES
鍵が受信側の秘密鍵を使用してアンラップされるか暗号化解除され、共用パーツが Triple-DES 鍵を使用し
て暗号化が解除されます。 m 個の共用パーツが受信されたときに、複製されたマスター鍵が、受信側の新
規のマスター鍵レジスター内で完成します。

最も簡単、かつ迅速にマスター鍵を複製するには、暗号化コプロセッサ構成のための Web ベースのユー
ティリティーを使用します。このユーティリティーには、マスター鍵複製アドバイザーが含まれています。
マスター鍵複製アドバイザーを開始するには、次のステップに従います。

1. 「暗号化コプロセッサの構成 (Cryptographic Coprocessor configuration)」 ページの「構成の管理」をク
リックします。

- | 2. 「マスター鍵」をクリックします。
- | 3. 装置を選択します。
- | 4. 有効なコプロセッサのプロファイルとパスワードを入力します。
- | 5. 「複製 (Clone)」ボタンをクリックします。

| 独自のアプリケーションを作成して、マスター鍵を複製することもできます。それには、次の API verb を
| 使用します。

- | • Cryptographic_Facility_Control (CSUACFC)
- | • PKA_Key_Token_Build (CSNDPKB) (アプリケーションの作成方法によっては、不要な場合もあります)
- | • PKA_Key_Generate (CSNDPKG)
- | • PKA_Public_Key_Register (CSNDPKR)
- | • One_Way_Hash (CSNBOWH)
- | • Digital_Signature_Generate (CSNDDSG)
- | • Master_Key_Distribution (CSUAMKD)

| プログラムの例

| 参考のために、9 つのペアのプログラム例が提供されています。各ペアには ILE C で作成されたプログラ
| ムと、ILE RPG で作成されたプログラムが含まれています。どちらのプログラムも実行する機能は同じで
| す。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

- | • 197 ページの『例: 暗号化コプロセッサでマスター鍵の共用パーツの最小値と最大値を設定するための
| ILE C プログラム』
- | • 199 ページの『例: 暗号化コプロセッサでマスター鍵の共用パーツの最小値と最大値を設定するための
| ILE RPG プログラム』
- | • 202 ページの『例: マスター鍵の複製のための保管鍵のペアを生成するための ILE C プログラム』
- | • 207 ページの『例: マスター鍵の複製のための保管鍵のペアを生成するための ILE RPG プログラム』
- | • 214 ページの『例: 公開鍵のハッシュを登録するための ILE C プログラム』
- | • 217 ページの『例: 公開鍵のハッシュを登録するための ILE RPG プログラム』
- | • 223 ページの『例: 公開鍵証明書を登録するための ILE C プログラム』
- | • 226 ページの『例: 公開鍵証明書を登録するための ILE RPG プログラム』
- | • 230 ページの『例: 公開鍵のトークンを認証するための ILE C プログラム』
- | • 235 ページの『例: 公開鍵のトークンを認証するための ILE RPG プログラム』
- | • 243 ページの『例: マスター鍵の共用パーツを取得するための ILE C プログラム』
- | • 246 ページの『例: マスター鍵の共用パーツを取得するための ILE RPG プログラム』
- | • 252 ページの『例: マスター鍵の共用パーツをインストールするための ILE C プログラム』
- | • 256 ページの『例: マスター鍵の共用パーツをインストールするための ILE RPG プログラム』

| プログラム例の残りの 2 つのペアは、マスター鍵の複製には必ずしも必要ではありません。ただし、前の
| プログラム例の開発とテスト用に役に立つかもしれません。

- | • 263 ページの『例: 保管鍵をリストするための ILE C プログラム』
- | • 265 ページの『例: 保管鍵をリストするための ILE RPG プログラム』

- 268 ページの『例: 保管鍵を削除するための ILE C プログラム』
- 270 ページの『例: 保管鍵を削除するための ILE RPG プログラム』

マスター・キーの複製については、「IBM PCI 暗号化コプロセッサ CCA 基本サービスのリファレンスおよびガイド (IBM PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide)」を参照してください。

例: 暗号化コプロセッサでマスター鍵の共用パーツの最小値と最大値を設定するための ILE C プログラム:

暗号化コプロセッサでマスター鍵の共用パーツの最小値と最大値を設定するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

/*-----*/
/* Set the M-of-N values in the Coprocessor. These values are */
/* used in cloning of the master key. The master key is      */
/* cryptographically split into N number of parts and M number of */
/* parts are needed to recover it.                             */
/*                                                             */
/* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 2000                */
/*                                                             */
/* This material contains programming source code for your    */
/* consideration. These examples have not been thoroughly    */
/* tested under all conditions. IBM, therefore, cannot       */
/* guarantee or imply reliability, serviceability, or function */
/* of these program. All programs contained herein are        */
/* provided to you "AS IS". THE IMPLIED WARRANTIES OF        */
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  */
/* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
/* these programs and files.                                   */
/*                                                             */
/* Note: Input format is more fully described in Chapter 2 of */
/*       IBM CCA Basic Services Reference and Guide           */
/*       (SC31-8609) publication.                             */
/*                                                             */
/* Parameters:                                                */
/* none.                                                       */
/*                                                             */
/* Example:                                                    */
/* CALL PGM(SETMOFN) PARM(5 15)                                */
/*                                                             */
/* Note: This program assumes the device to use                */
/* already identified either by defaulting to the CRP01       */
/* device or by being explicitly named using the              */
/* Cryptographic_Resource_Allocate verb. Also this           */
/* device must be varied on and you must be authorized       */
/* to use this device description.                             */
/*                                                             */
/* Use these commands to compile this program on the system:  */
/* ADDLIB LIB(QCCA)                                           */
/* CRTCMOD MODULE(SETMOFN) SRCFILE(SAMPLE)                    */
/* CRTPGM PGM(SETMOFN) MODULE(SETMOFN)                        */
/* BNDSRVPGM(QCCA/CSUACFC)                                    */
/*                                                             */
/* Note: Authority to the CSUACFC service program in the      */
/* QCCA library is assumed.                                   */
/*                                                             */
/* The Common Cryptographic Architecture (CCA) verb used is  */

```

```

| /* Cryptographic_Facilites_Control (CSUACFC).                */
| /*                                                            */
| /*-----*/
|
| #include "csucincl.h"    /* header file for CCA Cryptographic    */
|                          /* Service Provider                */
|
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
| #include "decimal.h"
|
| /*-----*/
| /* standard return codes                                     */
| /*-----*/
| #define ERROR    -1
| #define OK       0
| #define WARNING  4
|
| int main(int argc, char *argv[])
|
| {
|     /*-----*/
|     /* standard CCA parameters                               */
|     /*-----*/
|     long return_code = 0;
|     long reason_code = 0;
|     long exit_data_length = 2;
|     char exit_data[4];
|     char rule_array[2][8];
|     long rule_array_count = 2;
|
|     /*-----*/
|     /* fields unique to this sample program                 */
|     /*-----*/
|     decimal(15,5) mparm, nparm;
|     long verb_data[2];
|     long verb_data_length = 8;
|
|     /*-----*/
|     /* Process parameters.  Numeric parms from the command line are */
|     /* passed in decimal 15,5 format.  The parms need to be converted */
|     /* to int format.                                             */
|     /*-----*/
|     memcpy(&mparm,argv[1],sizeof(mparm));
|     memcpy(&nparm,argv[2],sizeof(nparm));
|     verb_data[0] = mparm;
|     verb_data[1] = nparm;
|
|     /*-----*/
|     /* Set keywords in the rule array                         */
|     /*-----*/
|     memcpy(rule_array,"ADAPTER1SET-MOFN", 16);
|
|     /*-----*/
|     /* Invoke the verb to set the M of N values              */
|     /*-----*/
|     CSUACFC( &return_code,
|             &reason_code,
|             &exit_data_length,
|             exit_data,
|             &rule_array_count,
|             (char *)rule_array,
|             &verb_data_length,
|             (unsigned char *)verb_data);

```

```

|
| /*-----*/
| /* Check the results of the call */
| /*-----*/
| if ( (return_code == OK) | (return_code == WARNING) )
| {
|   printf("M of N values were successfully set with ");
|   printf("return/reason codes %ld/%ld\n\n",
|         return_code, reason_code);
|   return(OK);
| }
| else
| {
|   printf("An error occurred while setting the M of N values.\n");
|   printf("Return/reason codes %ld/%ld\n\n",
|         return_code, reason_code);
|   return(ERROR);
| }
| }

```

例: 暗号化コプロセッサでマスター鍵の共用パーツの最小値と最大値を設定するための ILE RPG プログラム:

暗号化コプロセッサでマスター鍵の共用パーツの最小値と最大値を設定するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* SETMOFN
| D*
| D* Set the M-of-N values in the Cryptographic Coprocessor. These values
| D* are used in cloning of the master key. The master key is
| D* cryptographically split into N number of parts and M number of
| D* parts are needed to recover it.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D*       IBM CCA Basic Services Reference and Guide
| D*       (SC31-8609) publication.
| D*
| D* Parameters: M and N
| D*
| D* Example:
| D*   CALL PGM(SETMOFN) PARM(5 10)
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(SETMOFN) SRCFILE(SAMPLE)
| D* CRTPGM PGM(SETMOFN) MODULE(SETMOFN)
| D*       BNDDIR(QCCA/QC6BNDDIR)
| D*

```

```

D* Note: Authority to the CSUACFC service program in the
D*       QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Cryptographic_Facility_Control (CSUACFC)
D*
D*****
D*-----
D* Declare variables used on CCA SAPI calls
D*-----
D*          ** Return code
DRETURNCODE  S          9B 0
D*          ** Reason code
DREASONCODE  S          9B 0
D*          ** Exit data length
DEXITDATALEN S          9B 0
D*          ** Exit data
DEXITDATA    S          4
D*          ** Rule array count
DRULEARRAYCNT S        9B 0
D*          ** Rule array
DRULEARRAY   S          16
D*          ** Verb data length
DVERBDATALEN S          9B 0
D*          ** Verb data contain M (minimum) and N (maximum)
DVERBDATA    DS          8
DM            S          9B 0
DN            S          9B 0
D*
D*****
D* Prototype for Cryptographic_Facility_Control (CSUACFC)
D*****
DCSUACFC     PR
DRETCODE    S          9B 0
DRSNCODE    S          9B 0
DEXTDTALEN  S          9B 0
DEXTDTA     S          4
DRARRAYCT   S          9B 0
DRARRAY     S          16
DVRBDTALEN  S          9B 0
DVRBDTA     S          8
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG        S          75  DIM(2) CTDATA PERRCD(1)
MSGLENGTH   S          9B 0 INZ(75)
D           DS
DMSGTEXT    S          1    80
DFAILRETC   S          41   44
DFAILRSNC   S          46   49
DMESSAGEID  S          7    INZ(' ')
DMESSAGEFILE S          21  INZ(' ')
DMSGKEY     S          4    INZ(' ')
DMSGTYPE    S          10  INZ('*INFO ')
DSTACKENTRY S          10  INZ('* ')
DSTACKCOUNTER S        9B 0 INZ(2)
DERRCODE    DS
DBYTESIN    S          1    4B 0 INZ(0)
DBYTESOUT   S          5    8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM *
C*-----
C  *ENTRY      PLIST
C              PARM                MVALUE      15 5

```

```

C          PARM          NVALUE          15 5
C*-----*
C* Set the keyword in the rule array *
C*-----*
C          MOVE      'ADAPTER1'  RULEARRAY
C          MOVE      'SET-MOFN'  RULEARRAY
C          Z-ADD     2           RULEARRAYCNT
C*-----*
C* Set the verb data length to 8 *
C*-----*
C          Z-ADD     8           VERBDATALEN
C*-----*
C* Set the M and N value (Convert from decimal 15 5 to binary)*
C*-----*
C          EVAL      M = MVALUE
C          EVAL      N = NVALUE
C*****
C* Call Cryptographic Facility Control SAPI */
C*****
C          CALLP     CSUACFC      (RETURNCODE:
C                                REASONCODE:
C                                EXITDATALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                VERBDATALEN:
C                                VERBDATA)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      0
C*          *-----*
C*          * Send error message *
C*          *-----*
C          MOVE      MSG(1)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC
C          MOVE      REASONCODE  FAILRSNC
C          EXSR      SNDMSG
C*
C          ELSE
C*          *****
C*          * Send success message *
C*          *****
C          MOVE      MSG(2)      MSGTEXT
C          EXSR      SNDMSG
C*
C          ENDIF
C
C          SETON
C
C*****
C* Subroutine to send a message
C*****
C          SNDMSG      BEGSR
C          CALL        'QMHSNDPM'
C          PARM        MESSAGEID
C          PARM        MESSAGEFILE
C          PARM        MSGTEXT
C          PARM        MSGLENGTH
C          PARM        MSGTYPE
C          PARM        STACKENTRY
C          PARM        STACKCOUNTER
C          PARM        MSGKEY
C          PARM        ERRCODE

```

```

|      C                ENDSR
|  **
|  CSUACFC failed with return/reason codes 9999/9999.
|  The request completed successfully.

```

例: マスター鍵の複製のための保管鍵のペアを生成するための ILE C プログラム:

マスター鍵を複製するための保管鍵のペアを生成するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

|  /*-----*/
|  /* GENRETAIN */
|  /* */
|  /* Sample program to generate a retained key to be used for */
|  /* master key cloning. */
|  /* */
|  /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 1999 */
|  /* */
|  /* This material contains programming source code for your */
|  /* consideration. These examples have not been thoroughly */
|  /* tested under all conditions. IBM, therefore, cannot */
|  /* guarantee or imply reliability, serviceability, or function */
|  /* of these program. All programs contained herein are */
|  /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
|  /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
|  /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
|  /* these programs and files. */
|  /* */
|  /* */
|  /* Note: Input format is more fully described in Chapter 2 of */
|  /* IBM CCA Basic Services Reference and Guide */
|  /* (SC31-8609) publication. */
|  /* */
|  /* Parameters: RETAINED_KEY_NAME */
|  /* */
|  /* Example: */
|  /* CALL PGM(GENRETAIN) PARM(TESTKEY) */
|  /* */
|  /* */
|  /* Note: This program assumes the card with the profile is */
|  /* already identified either by defaulting to the CRP01 */
|  /* device or by being explicitly named using the */
|  /* Cryptographic_Resource_Allocate verb. Also this */
|  /* device must be varied on and you must be authorized */
|  /* to use this device description. */
|  /* */
|  /* The Common Cryptographic Architecture (CCA) verbs used are */
|  /* PKA_Key-Token_Build (CSNDPKB) and PKA_Key_Generate (CSNDPKG). */
|  /* */
|  /* Use these commands to compile this program on the system: */
|  /* ADDLIB LIB(QCCA) */
|  /* CRTCMOD MODULE(GENRETAIN) SRCFILE(SAMPLE) */
|  /* CRTPGM PGM(GENRETAIN) MODULE(GENRETAIN) */
|  /* BNDDIR(QCCA/QC6BNDDIR) */
|  /* */
|  /* Note: Authority to the CSNDPKG and CSNDPKB service programs */
|  /* in the QCCA library is assumed. */
|  /* */
|  /*-----*/
|  #include <stdio.h>
|  #include <string.h>
|  #include "csucincl.h"

```



```

| int main(int argc, char *argv[])
| {
| /*-----*/
| /* Declares for CCA parameters */
| /*-----*/
| long return_code = 0;
| long reason_code = 0;
| long exit_data_length = 0;
| char exit_data[4];
| char rule_array[24];
| long rule_array_count;
| long token_len = 2500;
| char token[2500];
| char regen_data[4];
| char transport_key_id[4];
| struct {
|     short modlen;
|     short modlenfld;
|     short pubexplen;
|     short prvexplen;
|     long pubexp;
| } key_struct; /* Key structure for PKA Key Token Build */
| long key_struct_length;
| long zero = 0;
| /*-----*/
| /* Declares for working with a PKA token */
| /*-----*/
| long pub_sec_len; /* Public section length */
| long prv_sec_len; /* Private section length */
| long cert_sec_len; /* Certificate section length */
| long info_subsec_len; /* Information subsection length */
| long offset; /* Offset into token */
| long tempOffset; /* (Another) Offset into token */
| long tempLength; /* Length variable */
| long tempLen1, tempLen2; /* temporary length variables */
| char pub_token[2500];
| long pub_token_len;
| long name_len;
| char name[64];
|
| int i; /* Loop counter */
| FILE *fp; /* File pointer */
|
| if (argc < 2) /* Check the number of parameters passed */
| {
|     printf("Need to enter a private key name\n");
|     return 1;
| }
|
| memset(token,0,2500); /* Initialize token to 0 */
| memcpy((void*)rule_array,"RSA-PRIVKEY-MGMT",16); /* Set rule array */
| rule_array_count = 2;
|
| memset(name, ' ', 64); /* Copy key name parameter */
| memcpy(name, argv[1], strlen(argv[1]));
| name_len = 64;
|
| /*-----*/
| /* Initialize key structure */
| /*-----*/
| memset((void*)&key_struct, 0, sizeof(key_struct));
| key_struct.modlen = 1024; /* Modulus length is 1024 */
| key_struct.pubexplen = 3;
| key_struct.pubexp = 0x01000100; /* Public exponent is 65537 */
| key_struct_length = sizeof(key_struct);

```

```

| /*****
| /* Call PKA_Key_Token_Build SAPI */
| /*****
| CSNDPKB( &return_code, &reason_code, &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         rule_array,
|         &key_struct_length,
|         (unsigned char *)&key_struct,
|         &name_len,
|         name,
|         &zero,          /* 1 */
|         NULL,
|         &zero,          /* 2 */
|         NULL,
|         &zero,          /* 3 */
|         NULL,
|         &zero,          /* 4 */
|         NULL,
|         &zero,          /* 5 */
|         NULL,
|         &token_len,
|         token);
|
| if (return_code != 0)
| {
|     printf("PKA Key Token Build Failed : return code %d : reason code %d\n",
|           return_code, reason_code);
|     return 1;
| }
|
| /*****
| /* Build certificate */
| /*****
|         /* Determine length of token from length */
|         /* bytes at offset 2 and 3. */
| token_len = ((256 * token[2]) + token[3]);
|         /* Determine length of private key */
|         /* section from length bytes at offset */
|         /* 10. */
| prv_sec_len = ((256 * token[10]) + token[11]);
|         /* Determine length of public key section*/
|         /* section from length bytes at offset */
|         /* 10 + private section length */
| pub_sec_len = ((256 * token[prv_sec_len + 10]) +
|               token[prv_sec_len + 11]);
|
|         /* Calculate the signature section length*/
| cert_sec_len = 328 + /* from the signature subsection length, */
|                 20 + /* EID subsection length, */
|                 12 + /* Serial number subsection length, */
|                 4 + /* Information subsection header length, */
|                 pub_sec_len + /* Public key subsection length, */
|                 4; /* and the certificate section hdr length*/
|
| offset = token_len; /* Offset for additions to token */
|
| /* Fill in certificate section header */
| tempLen1 = cert_sec_len;
| tempLen1 >>= 8;
| token[offset++] = 0x40;
| token[offset++] = 0x00;
| token[offset++] = tempLen1;
| token[offset++] = cert_sec_len;
|
| /* Fill in public key subsection */
| token[offset++] = 0x41;

```

```

|   for (i = 1 ; i < pub_sec_len ; i ++)
|   {
|       /* Copy public key to certificate */
|       token[offset++] = token[prv_sec_len +(i+8)];
|   }
|
|   /* Fill Optional Information Subsection Header */
|   info_subsec_len = 20 + /* Length of EID section          */
|                       12 + /* Length of serial number section */
|                       4; /* Length of Info subsection header */
|   tempLen1 = info_subsec_len;
|   tempLen1 >>= 8;
|   token[offset++] = 0x42;
|   token[offset++] = 0x00;
|   token[offset++] = tempLen1;
|   token[offset++] = info_subsec_len;
|
|   /* Fill in Public Key Certificate EID subsection */
|   token[offset++] = 0x51;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x14;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|
|   /* Public key Certificate Serial Number TLV */
|   token[offset++] = 0x52;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x0c;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|   token[offset++] = 0x00;
|
|   /* Fill in Signature Subsection */
|   token[offset++] = 0x45;
|   token[offset++] = 0x00;
|   token[offset++] = 0x01;
|   token[offset++] = 0x48;
|   token[offset++] = 0x01;
|   token[offset++] = 0x01;
|
|   for (i = 0 ; i < 64 ;i++)
|   {
|       /* Copy private key name out of private key name section */
|       /* into certificate */
|       token[offset++] =
|           token[prv_sec_len + pub_sec_len + 12 + i];

```

```

    }

    token_len = offset + 258; /* add 258 to allow for digital sig. */
    token[3] = token_len; /* Set new token length */
    token[2] = token_len >> 8;

    /******
    /* Generate Retained key using PKA token with certificate */
    /******
    memcpy((void*)rule_array,"RETAIN CLONE ",16);
    rule_array_count = 2;
    memset(pub_token,0,2500);
    pub_token_len = 2500;
    memset(transport_key_id,0,4);

    /******
    /* Call PKA_Key_Generate SAPI */
    /******
    CSNDPKG( &return_code, &reason_code, &exit_data_length,
            exit_data,
            &rule_array_count,
            rule_array,
            &zero, /* regenerated data length */
            regen_data,
            &token_len,
            token,
            transport_key_id,
            &pub_token_len,
            pub_token);

    if (return_code != 0)
    {
        printf("PKA Key Generate Failed : return code %d :reason code %d\n",
            return_code, reason_code);
        return 1;
    }

    /******
    /* Write public key token out to file */
    /******
    /* Append ".PUB" to key name */
    memcpy((void*)&name[strlen(argv[1])],".PUB",5);
    fp = fopen(name,"wb"); /* Open the file */

    if (!fp)
    {
        printf("File open failed\n");
    }
    else
    {
        fwrite(pub_token,pub_token_len,1,fp); /* Write token to file */

        fclose(fp); /* Close the file */
        printf("Public token written to file %s.\n",name);
    }

    name[strlen(argv[1])] = 0; /* Convert name to string */
    printf("Private key %s is retained in the hardware\n",name);
    return 0;
}

```

| 例: マスター鍵の複製のための保管鍵のペアを生成するための ILE RPG プログラム:

| マスター鍵を複製するための保管鍵のペアを生成するには、必要に応じて以下のプログラム例を変更してく
| ださい。

| 注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』
| を参照してください。

```
| D*****  
| D* GENRETAIN  
| D*  
| D* Sample program to generate a retained key to be used for  
| D* master key cloning.  
| D*  
| D*  
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000  
| D*  
| D* This material contains programming source code for your  
| D* consideration. These example has not been thoroughly  
| D* tested under all conditions. IBM, therefore, cannot  
| D* guarantee or imply reliability, serviceability, or function  
| D* of these programs. All programs contained herein are  
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF  
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE  
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for  
| D* these programs and files.  
| D*  
| D*  
| D* Note: Input format is more fully described in Chapter 2 of  
| D* IBM CCA Basic Services Reference and Guide  
| D* (SC31-8609) publication.  
| D*  
| D* Parameters: RETAINED_KEY_NAME  
| D*  
| D* Example:  
| D* CALL PGM(GENRETAIN) PARM(TESTKEY)  
| D*  
| D* Use these commands to compile this program on the system:  
| D* CRTRPGMOD MODULE(GENRETAIN) SRCFILE(SAMPLE)  
| D* CRTPGM PGM(GENRETAIN) MODULE(GENRETAIN)  
| D* BNDDIR(QCCA/QC6BNDDIR)  
| D*  
| D* Note: Authority to the CSNDPKG and CSNDPKB service programs  
| D* in the QCCA library is assumed.  
| D*  
| D* The Common Cryptographic Architecture (CCA) verbs used are  
| D* PKA_Key_Token_Build (CSNDPKB) and PKA_Key_Generate (CSNDPKG).  
| D*  
| D*****  
| D*-----  
| D* Declare variables used by CCA SAPI calls  
| D*-----  
| D* ** Return code  
| DRETURNCODE S 9B 0  
| D* ** Reason code  
| DREASONCODE S 9B 0  
| D* ** Exit data length  
| DEXITDATALEN S 9B 0  
| D* ** Exit data  
| DEXITDATA S 4  
| D* ** Rule array count  
| DRULEARRAYCNT S 9B 0  
| D* ** Rule array  
| DRULEARRAY S 16  
| D* ** Token length  
| DTOKENLEN S 9B 0 INZ(2500)
```

```

|      D*          ** Token and array for subscripting
|      DTOKEN      DS          2500
|      DTOKENARRAY      1      DIM(2500)
|      D*          ** Regeneration data
|      DREGENDATA    S          4      INZ(X'00000000')
|      D*          ** Transport key encrypting key
|      DTRANSPORTKEK S          4      INZ(X'00000000')
|      D*          ** Generated keyid
|      DGENKEY       S          2500
|      D*          ** Generated keyid length
|      DGENKEYLEN    S          9B 0 INZ(2500)
|      D*          ** Key name and length
|      DKEYNAME      S          64
|      DKEYNAMEL     S          9B 0 INZ(64)
|      D*          ** Key structure for PKA Key Token Build
|      DKEYSTRUCT    DS
|      DMODLEN       1          2B 0
|      DMODLENFLD    3          4B 0
|      DPUBEXPLEN    5          6B 0
|      DPRVEXPLEN    7          8B 0
|      DPUBEXP       9          12B 0
|      D*          ** Null parms needed for CSNDPKB and CSNDPKG
|      DZERO         S          9B 0 INZ(0)
|      DNULLPTR      S          *      INZ(*NULL)
|      D*          ** Key structure length
|      DKEYSTRUCTLEN S          9B 0 INZ(12)
|      D*          ** Data structure for aligning 2 bytes into
|      D*          ** a 2 bytes integer
|      DLENSTRUCT    DS          2
|      DMSB          1          1
|      DLSB          2          2
|      DLENGTH       1          2B 0
|      D*          ** Private key section length
|      DPRVSECLLEN  S          9B 0
|      D*          ** Public key section length
|      DPUBSECLLEN  S          9B 0
|      D*          ** Index into Token array
|      DINDEX       S          9B 0
|      D*          ** Declares for copying private key name
|      DNAMEPTR1    S          *
|      DNAME1       S          64      BASED(NAMEPTR1)
|      DNAMEPTR2    S          *
|      DNAME2       S          64      BASED(NAMEPTR2)
|      D*          ** Loop counter
|      DI           S          9B 0
|      D*          ** File descriptor
|      DFILED      S          9B 0
|      D*          ** File path and length
|      DPATH       S          80      INZ(*ALLX'00')
|      DPATHLEN    S          9B 0
|      D*          ** Open flag - Create on open, open for writing,
|      D*          ** and clear if exists
|      DOFLAG      S          10I 0 INZ(X'4A')
|      D*
|      D*****
|      D* Prototype for PKA_Key-Token_Build (CSNDPKB)
|      D*****
|      DCSNDPKB     PR
|      DRETCODE     9B 0
|      DRSNCODE     9B 0
|      DEXTDTALEN   9B 0
|      DEXTDTA      4
|      DRARRAYCT    9B 0
|      DRARRAY      16
|      DKEYSTRLEN   9B 0
|      DKEYSTR      10
|      DKEYNML      9B 0

```

```

| DKEYNM                64
| DRSRVLN1              9B 0
| DRSRV1                *  VALUE
| DRSRVLN2              9B 0
| DRSRV2                *  VALUE
| DRSRVLN3              9B 0
| DRSRV3                *  VALUE
| DRSRVLN4              9B 0
| DRSRV4                *  VALUE
| DRSRVLN5              9B 0
| DRSRV5                *  VALUE
| DTKNLEN              9B 0
| DTKN                  2500  OPTIONS(*VARSIZE)
| D*
| D*****
| D* Prototype for PKA_Key_Generate (CSNDPKG)
| D*****
| DCSNDPKG              PR
| DRETCOD                9B 0
| DRSNCOD                9B 0
| DEXTDTALN             9B 0
| DEXTDT                 4
| DRARRYCT              9B 0
| DRARRY                 16
| DREGDTAL              9B 0
| DREGDTA                20  OPTIONS(*VARSIZE)
| DSKTKNL               9B 0
| DSKTKN                 2500  OPTIONS(*VARSIZE)
| DTRNKEK                64  OPTIONS(*VARSIZE)
| DGENKEYL              9B 0
| DGENKEY                2500  OPTIONS(*VARSIZE)
| D*
| D*****
| D* Prototype for open()
| D*****
| D* value returned = file descriptor (OK), -1 (error)
| Dopen                 PR          9B 0  EXTPROC('open')
| D* path name of file to be opened.
| D                     128  OPTIONS(*VARSIZE)
| D* Open flags
| D                     9B 0  VALUE
| D* (OPTIONAL) mode - access rights
| D                     10U 0  VALUE  OPTIONS(*NOPASS)
| D* (OPTIONAL) codepage
| D                     10U 0  VALUE  OPTIONS(*NOPASS)
| D*
| D*****
| D* Prototype for write()
| D*****
| D* value returned = number of bytes actually written, or -1
| Dwrite                PR          9B 0  EXTPROC('write')
| D* File descriptor returned from open()
| D                     9B 0  VALUE
| D* Data to be written
| D                     1200  OPTIONS(*VARSIZE)
| D* Length of data to write
| D                     9B 0  VALUE
| D*
| D*****
| D* Prototype for close()
| D*****
| D* value returned = 0 (OK), or -1
| Dclose                PR          9B 0  EXTPROC('close')
| D* File descriptor returned from open()
| D                     9B 0  VALUE
| D*
| D*-----

```

```

D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG          S          75    DIM(4) CTDATA PERRCD(1)
DMSGLENGTH   S          9B 0  INZ(75)
D
DMSGTEXT      1          75
DSAPI         1          7
DFAILRETC    41         44
DFAILRSNC    46         49
DMESSAGEID   S          7    INZ('      ')
DMESSAGEFILE S          21   INZ('      ')
DMSGKEY      S          4    INZ('      ')
DMSGTYPE     S          10   INZ('*INFO ')
DSTACKENTRY  S          10   INZ('*      ')
DSTACKCOUNTER S         9B 0  INZ(2)
DERRCODE     DS
DBYTESIN     1          4B 0  INZ(0)
DBYTESOUT    5          8B 0  INZ(0)
C*
C*****
C* START OF PROGRAM *
C* *
C   *ENTRY      PLIST
C               PARM                KEYNAMEPARM    50
C* *-----*
C* * Initialize tokens to 0 *
C* *-----*
C               MOVEL   *ALLX'00'    TOKEN
C               MOVEL   *ALLX'00'    GENKEY
C* *-----*
C* * Initialize key struct *
C* *-----*
C               Z-ADD   1024          MODLEN
C               Z-ADD   0            MODLENFLD
C               Z-ADD   3            PUBEXPLEN
C               Z-ADD   0            PRVEXPLEN
C               EVAL    PUBEXP = 65537 * 256
C* *-----*
C* * Copy key name from parm*
C* *-----*
C               MOVEL   KEYNAMEPARM   KEYNAME
C* *-----*
C* * Set the keywords in the rule array *
C* *-----*
C               MOVEL   'RSA-PRIV'    RULEARRAY
C               MOVE    'KEY-MGMT'    RULEARRAY
C               Z-ADD   2            RULEARRAYCNT
C*****
C* Call PKA_Key_Token_Build SAPI
C*****
C               CALLP   CSNDPKB      (RETURNCODE:
C                                   REASONCODE:
C                                   EXITDATALEN:
C                                   EXITDATA:
C                                   RULEARRAYCNT:
C                                   RULEARRAY:
C                                   KEYSTRUCTLEN:
C                                   KEYSTRUCT:
C                                   KEYNAMELEN:
C                                   KEYNAME:
C                                   ZERO:
C                                   NULLPTR:
C                                   ZERO:
C                                   NULLPTR:
C                                   ZERO:
C                                   NULLPTR:

```



```

C                                     ZERO:
C                                     NULLPTR:
C                                     ZERO:
C                                     NULLPTR:
C                                     TOKENLEN:
C                                     TOKEN)
C*  *-----*
C*  * Check the return code *
C*  *-----*
C    RETURNCODE    IFGT    0
C*  *-----*
C*  * Send failure message *
C*  *-----*
C          MOVE    MSG(1)    MSGTEXT
C          MOVE    RETURNCODE  FAILRETC
C          MOVE    REASONCODE  FAILRSNC
C          MOVE    'CSNDPKB'   SAPI
C          EXSR    SNDMSG
C          RETURN
C          ENDIF
C*
C*-----*
C* Build the certificate *
C*-----*
C*  Get the private section length. The length is at position 11
C*  of the token
C          EVAL    MSB = TOKENARRAY(10+1)
C          EVAL    LSB = TOKENARRAY(11+1)
C          MOVE    LENGTH    PRVSECLN
C*  Get the public section length. The length is at position
C*  (11 + Private key section length).
C          EVAL    MSB = TOKENARRAY(10 + PRVSECLN + 1)
C          EVAL    LSB = TOKENARRAY(11 + PRVSECLN + 1)
C          MOVE    LENGTH    PUBSECLN
C*  Calculate the certificate section length
C*  Cert Section length = Signature length (328) +
C*  EID section length (20) +
C*  Serial number length (12) +
C*  Info subsection header length (4) +
C*  Public Key section length +
C*  Cert section header length (4)
C          EVAL    LENGTH = 328 + 20 + 12 + 4 + PUBSECLN + 4
C*  Fill Certificate section header
C          MOVE    TOKENLEN    INDEX
C          EVAL    TOKENARRAY(INDEX +1) = X'40'
C          EVAL    TOKENARRAY(INDEX +2) = X'00'
C          EVAL    TOKENARRAY(INDEX +3) = MSB
C          EVAL    TOKENARRAY(INDEX +4) = LSB
C*  Fill in public key subsection
C          EVAL    TOKENARRAY(INDEX +5) = X'41'
C          ADD     5          INDEX
C          Z-ADD   1          I
C*  Copy the public key section of the token into the public key
C*  subsection of the certificate section.
C    I          DOWLT    PUBSECLN
C          EVAL    TOKENARRAY(INDEX + I) =
C          TOKENARRAY(PRVSECLN + I + 8 + 1)
C    1          ADD     I          I
C          ENDDO
C          EVAL    INDEX = INDEX + PUBSECLN - 1
C*  Fill in Optional Information subsection header
C          Z-ADD   36          LENGTH
C          EVAL    TOKENARRAY(INDEX +1) = X'42'
C          EVAL    TOKENARRAY(INDEX +2) = X'00'
C          EVAL    TOKENARRAY(INDEX +3) = MSB
C          EVAL    TOKENARRAY(INDEX +4) = LSB
C*  Fill in Public Key Certificate EID

```

```

C          EVAL      INDEX = INDEX + 4
C          EVAL      TOKENARRAY(INDEX +1) = X'51'
C          EVAL      TOKENARRAY(INDEX +4) = X'14'
C*   Fill in Public Key Certificate Serial Number TLV
C          EVAL      INDEX = INDEX + 20
C          EVAL      TOKENARRAY(INDEX +1) = X'52'
C          EVAL      TOKENARRAY(INDEX +4) = X'0C'
C*   Fill in Signature Subsection
C          EVAL      INDEX = INDEX + 12
C          EVAL      TOKENARRAY(INDEX +1) = X'45'
C          EVAL      TOKENARRAY(INDEX +3) = X'01'
C          EVAL      TOKENARRAY(INDEX +4) = X'48'
C          EVAL      TOKENARRAY(INDEX +5) = X'01'
C          EVAL      TOKENARRAY(INDEX +6) = X'01'
C*   Fill in private key name
C          EVAL      INDEX = INDEX + 6
C          EVAL      NAMEPTR1 = %ADDR(TOKENARRAY(INDEX +1))
C          EVAL      NAMEPTR2 =
C                  %ADDR(TOKENARRAY(PRVSECLN+PUBSECLN+12+1))
C          MOVE      NAME2      NAME1
C*   Adjust token length
C          EVAL      LENGTH = INDEX + 64 + 258
C          MOVE      MSB      TOKENARRAY(3)
C          MOVE      LSB      TOKENARRAY(4)
C          EVAL      TOKENLEN = LENGTH
C*   *-----*
C*   * Set the keywords in the rule array *
C*   *-----*
C          MOVE      'RETAIN '      RULEARRAY
C          MOVE      'CLONE '      RULEARRAY
C          Z-ADD      2      RULEARRAYCNT
C
C*-----*
C* Call PKA_Key_Generate SAPI *
C*-----*
C          CALLP      CSNDPKG      (RETURNCODE:
C                                REASONCODE:
C                                EXITDATALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                ZERO:
C                                REGENDATA:
C                                TOKENLEN:
C                                TOKEN:
C                                TRANSPORTKEK:
C                                GENKEYLEN:
C                                GENKEY)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      0
C*   *-----*
C*   * Send failure message *
C*   *-----*
C          MOVE      MSG(1)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC
C          MOVE      REASONCODE  FAILRSNC
C          MOVE      'CSNDPKG'   SAPI
C          EXSR      SNDMSG
C          RETURN
C          ENDIF
C*
C*   *-----*
C*   * Send success message *
C*   *-----*
C          MOVE      MSG(2)      MSGTEXT

```

```

C          EXSR      SNDMSG
C*
C*-----*
C* Write certificate out to file *
C*-----*
C*  ** Build path name
C          EVAL      PATHLEN = %LEN(%TRIM(KEYNAMEPARM))
C  PATHLEN      SUBST  KEYNAMEPARM:1 PATH
C          EVAL      %SUBST(PATH:PATHLEN+1:4) = '.PUB'
C*
C*  ** Open the file
C*
C          EVAL      FILED = open(PATH: OFLAG)
C*
C*  ** Check if open worked
C*
C  FILED      IFEQ      -1
C*
C*  ** Open failed, send an error message
C*
C          MOVE      MSG(3)      MSGTEXT
C          EXSR      SNDMSG
C*
C          ELSE
C*
C*  ** Open worked, write certificate out to file and close file
C*
C          CALLP      write      (FILED:
C                                GENKEY:
C                                GENKEYLEN)
C          CALLP      close      (FILED)
C*
C*  ** Send completion message
C*
C          MOVE      MSG(4)      MSGTEXT
C          EVAL      %SUBST(MSGTEXT: 32: PATHLEN + 4) =
C                                %SUBST(PATH: 1: PATHLEN + 4)
C          EXSR      SNDMSG
C          ENDIF
C*
C          SETON                                  LR
C*
C*****
C* Subroutine to send a message
C*****
C  SNDMSG      BEGSR
C          CALL      'QMHSNDPM'
C          PARM      MESSAGEID
C          PARM      MESSAGEFILE
C          PARM      MSGTEXT
C          PARM      MSGLENGTH
C          PARM      MSGTYPE
C          PARM      STACKENTRY
C          PARM      STACKCOUNTER
C          PARM      MSGKEY
C          PARM      ERRCODE
C          ENDSR
C*
**
| CSNDPKB failed with return/reason codes 9999/9999.
| The retained key was successfully created.
| The file could not be opened.
| The certificate was written to

```

例: 公開鍵のハッシュを登録するための ILE C プログラム:

公開鍵の証明書のハッシュを登録するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```
/*-----*/
/* REGHASH */
/* */
/* Sample program to register the hash of a CCA public key */
/* certificate. */
/* */
/* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 1999 */
/* */
/* This material contains programming source code for your */
/* consideration. These examples have not been thoroughly */
/* tested under all conditions. IBM, therefore, cannot */
/* guarantee or imply reliability, serviceability, or function */
/* of these program. All programs contained herein are */
/* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
/* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
/* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
/* these programs and files. */
/* */
/* Note: Input format is more fully described in Chapter 2 of */
/* IBM CCA Basic Services Reference and Guide */
/* (SC31-8609) publication. */
/* */
/* Parameters: Stream file containing public key certificate */
/* */
/* Example: */
/* CALL PGM(REGHASH) PARM(CERTFILE) */
/* */
/* Note: This program assumes the card with the profile is */
/* already identified either by defaulting to the CRP01 */
/* device or by being explicitly named using the */
/* Cryptographic_Resource_Allocate verb. Also this */
/* device must be varied on and you must be authorized */
/* to use this device description. */
/* */
/* The Common Cryptographic Architecture (CCA) verbs used are */
/* PKA_Public_Key_Hash_Register (CSNDPKH) and One_Way_Hash WH). */
/* (CSNBOWH). */
/* */
/* Use these commands to compile this program on the system: */
/* ADDLIB LIB(QCCA) */
/* CRTCMOD MODULE(REGHASH) SRCFILE(SAMPLE) */
/* CRTPGM PGM(REGHASH) MODULE(REGHASH) */
/* BNDDIR(QCCA/QC6BNDDIR) */
/* */
/* Note: Authority to the CSNDPKH and CSNBOWH service programs */
/* in the QCCA library is assumed. */
/* */
/*-----*/
#include <stdio.h>
#include <string.h>
#include "csucinl.h"

int main(int argc, char *argv[])
{
/*-----*/
/* Declares for CCA parameters */
/*-----*/
}
```

```

| long return_code = 0;
| long reason_code = 0;
| long exit_data_length = 0;
| char exit_data[4];
| char rule_array[24];
| long rule_array_count;
| long token_len = 2500;
| char token[2500];
| long chaining_vector_length = 128;
| long hash_length = 20;
| long text_length;
| unsigned char chaining_vector[128];
| unsigned char hash[20];
| /*-----*/
| /* Declares for working with a PKA token */
| /*-----*/
| long pub_sec_len; /* Public section length */
| long cert_sec_len; /* Certificate section length */
| long offset; /* Offset into token */
| long tempOffset; /* (Another) Offset into token */
| char name[64]; /* Registered key name */
|
| long count; /* Number of bytes read from file */
| FILE *fp; /* File pointer */
|
| if (argc < 2) /* Check the number of parameters passed */
| {
|     printf("Need to enter a public key name\n");
|     return 1;
| }
|
| memset(name, ' ', 64); /* Copy key name (and pad) to a 64 byte */
| /* field. */
| memcpy(name, argv[1], strlen(argv[1]));
|
| fp = fopen(argv[1], "rb"); /* Open the file for reading */
| if (!fp)
| {
|     printf("File %s not found.\n", argv[1]);
|     return 1;
| }
|
| memset(token, 0, 2500); /* Initialize the token to 0 */
| count = fread(token, 1, 2500, fp); /* Read the token from the file */
| fclose(fp); /* Close the file */
|
| /* Determine length of token from length */
| /* bytes at offset 2 and 3. */
| token_len = ((256 * token[2]) + token[3]);
| if (count < token_len) /* Check if whole token was read in */
| {
|     printf("Incomplete token in file\n");
|     return 1;
| }
|
| /*****/
| /* Find the certificate offset in the token */
| /*
| /* The layout of the token is
| /*
| /* - Token header - 8 bytes - including 2 length bytes
| /* - Public key section - length bytes at offset 10 overall
| /* - Private key name - 68 bytes
| /* - Certificate section
| /*
| /*****/
| pub_sec_len = ((256 * token[10]) + token[11]);

```

```

offset = pub_sec_len + 68 + 8; /* Set offset to certificate section */

/* Determine certificate section */
/* length from the length bytes at */
/* offset 2 of the section. */
cert_sec_len = ((256 * token[offset + 2]) + token[offset + 3]);
tempOffset = offset + 4; /* Set offset to first subsection */

/*-----*/
/* Parse each subsection of the certificate until the */
/* signature subsection is found or the end is reached.*/
/* (Identifier for signature subsection is Hex 45.) */
/*-----*/
while(token[tempOffset] != 0x45 &&
tempOffset < offset + cert_sec_len)
{
tempOffset += 256 * token[tempOffset + 2] + token[tempOffset+3];
}

/*-----*/
/* Check if no signature was found before the end of */
/* the certificate section. */
/*-----*/
if (token[tempOffset] != 0x45)
{
printf("Invalid certificate\n");
return 1;
}

/*****
/* Hash the certificate */
/*****
text_length = tempOffset - offset + 70; /* Text length is length */
/* of certificate subsection. */

memcpy((void*)rule_array,"SHA-1 ",8); /* Set rule array */
rule_array_count = 1;
chaining_vector_length = 128;
hash_length = 20;

CSNBOWH( &return_code, &reason_code, &exit_data_length,
exit_data,
&rule_array_count,
(unsigned char*)rule_array,
&text_length,
&token[offset],
&chaining_vector_length,
chaining_vector,
&hash_length,
hash);

if (return_code != 0)
{
printf("One_Way_Hash Failed : return reason %d/%d\n",
return_code, reason_code);
return 1;
}

/*****
/* Register the Hash */
/*****
/* Set the rule array */
memcpy((void*)rule_array,"SHA-1 CLONE ",16);
rule_array_count = 2;

/* Build the name of the retained */
/* key from the file and "RETAINED"*/

```

```

| memcpy(&name[strlen(argv[1])],".RETAINED",9);
|
| CSNDPKH( &return_code, &reason_code, &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char*)rule_array,
|         name,
|         &hash_length,
|         hash);
|
| if (return_code != 0)
| {
|     printf("Public Key Register_Hash Failed : return reason %d/%d¥n",
|           return_code, reason_code);
|     return 1;
| }
|
| name[strlen(argv[1]) + 9] = 0; /* Convert name to a string          */
| printf("Hash registered for %s.¥n",name);
|
| }

```

例: 公開鍵のハッシュを登録するための ILE RPG プログラム:

公開鍵の証明書のハッシュを登録するには、必要に応じて以下のプログラム例を変更してください。

公開鍵の証明書のハッシュを登録するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* REGHASH
| D*
| D* Sample program to register the hash of a CCA public key
| D* certificate.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D*       IBM CCA Basic Services Reference and Guide
| D*       (SC31-8609) publication.
| D*
| D* Parameters: Stream file containing public key certificate
| D*
| D* Example:
| D*   CALL PGM(REGHASH) PARM(CERTFILE)
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(REGHASH) SRCFILE(SAMPLE)
| D* CRTPGM PGM(REGHASH) MODULE(REGHASH)
| D*       BNDDIR(QCCA/QC6BNDDIR)
| D*
| D* Note: Authority to the CSNDPKH and CSNBOWH service programs

```

```

|      D*      in the QCCA library is assumed.
|      D*
|      D* The Common Cryptographic Architecture (CCA) verbs used are
|      D* PKA_Public_Key_Hash_Register (CSNDPKH) and One_Way_Hash
|      C* (CSNBOWH).
|      D*
|      D*****
|      D*-----
|      D* Declare variables used by CCA SAPI calls
|      D*-----
|      D*      ** Return code
|      DRETURNCODE      S          9B 0
|      D*      ** Reason code
|      DREASONCODE      S          9B 0
|      D*      ** Exit data length
|      DEXITDATALEN      S          9B 0
|      D*      ** Exit data
|      DEXITDATA        S          4
|      D*      ** Rule array count
|      DRULEARRAYCNT     S          9B 0
|      D*      ** Rule array
|      DRULEARRAY        S          16
|      D*      ** Token length
|      DTOKENLEN         S          9B 0 INZ(2500)
|      D*      ** Token and array for subscribing token
|      DTOKEN            DS         2500
|      DTOKENARRAY       S          1 DIM(2500)
|      D*      ** Chaining vector length
|      DCHAINVCTLEN      S          9B 0 INZ(128)
|      D*      ** Chaining vector
|      DCHAINVCT         S          128
|      D*      ** Hash length
|      DHASHLEN          S          9B 0 INZ(20)
|      D*      ** Hash
|      DHASH             S          20
|      D*      ** Text length
|      DTXTLENGTH        S          9B 0
|      D*      ** Name of retained key
|      DNAME             S          64
|      D*      ** Structure used for aligning 2 bytes into a
|      D*      ** 2 byte integer.
|      DLENSTRUCT        DS         2
|      DMSB              S          1 1
|      DLSB              S          2 2
|      DLENGTH           S          1 2B 0
|      D*
|      D*      ** Certificate section length
|      DCRTSECLen        S          9B 0
|      D*      ** Public key section length
|      DPUBSECLen        S          9B 0
|      D*      ** Index into PKA key token
|      DTKNINDEX         S          9B 0
|      D*      ** Index into PKA key token
|      DTMPINDEX         S          9B 0
|      D*      ** File descriptor
|      DFILED            S          9B 0
|      D*      ** File path and path length
|      DPATH             S          80 INZ(*ALLX'00')
|      DPATHLEN          S          9B 0
|      D*      ** Open Flag - Open for Read only
|      DOFLAG            S          10I 0 INZ(1)
|      D*
|      D*****
|      D* Prototype for PKA_Public_Key_Hash_Register (CSNDPKH)
|      D*****
|      DCSNDPKH          PR
|      DRETCOD           S          9B 0

```



```

|      DRSNCOD                9B 0
|      DEXTDTALN              9B 0
|      DEXTDT                  4
|      DRARRYCT               9B 0
|      DRARRY                  16
|      DKYNAM                  64
|      DSHL                    9B 0
|      DSH                      20  OPTIONS(*VARSIZE)
|      D*
|      D*****
|      D* Prototype for One_Way_Hash (CSNBOWH)
|      D*****
|      DCSNBOWH                PR
|      DRETCOD                 9B 0
|      DRSNCOD                 9B 0
|      DEXTDTALN              9B 0
|      DEXTDT                  4
|      DRARRYCT               9B 0
|      DRARRY                  16
|      DTXTLEN                 9B 0
|      DTXT                     500  OPTIONS(*VARSIZE)
|      DCHNVCTLEN              9B 0
|      DCHNVCT                 128
|      DSHLEN                  9B 0
|      DSH                      20
|      D*
|      D*
|      D*****
|      D* Prototype for open()
|      D*****
|      D*   value returned = file descriptor (OK), -1 (error)
|      Dopen                    PR          9B 0  EXTPROC('open')
|      D*   path name of file to be opened.
|      D                        128  OPTIONS(*VARSIZE)
|      D*   Open flags
|      D                        9B 0  VALUE
|      D*   (OPTIONAL) mode - access rights
|      D                        10U 0  VALUE OPTIONS(*NOPASS)
|      D*   (OPTIONAL) codepage
|      D                        10U 0  VALUE OPTIONS(*NOPASS)
|      D*
|      D*****
|      D* Prototype for read()
|      D*****
|      D*   value returned = number of bytes actually read, or -1
|      Dread                    PR          9B 0  EXTPROC('read')
|      D*   File descriptor returned from open()
|      D                        9B 0  VALUE
|      D*   Input buffer
|      D                        2500  OPTIONS(*VARSIZE)
|      D*   Length of data to be read
|      D                        9B 0  VALUE
|      D*
|      D*****
|      D* Prototype for close()
|      D*****
|      D*   value returned = 0 (OK), or -1
|      Dclose                   PR          9B 0  EXTPROC('close')
|      D*   File descriptor returned from open()
|      D                        9B 0  VALUE
|      D*
|      D*-----
|      D*           ** Declares for sending messages to the
|      D*           ** job log using the QMHSDPM API
|      D*-----
|      DMSG                      S          75  DIM(6) CTDATA PERRCD(1)
|      DMSGLENGTH                S          9B 0  INZ(75)

```

```

D          DS
DMSGTEXT      1      80
DSAPI         1      7
DFAILRETC    41     44
DFAILRSNC    46     49
DMESSAGEID   S       7      INZ('      ')
DMESSAGEFILE S      21     INZ('          ')
DMSGKEY      S       4      INZ('      ')
DMSGTYPE     S      10     INZ('*INFO  ')
DSTACKENTRY  S      10     INZ('*      ')
DSTACKCOUNTER S     9B 0   INZ(2)
DERRCODE     DS
DBYTESIN     1      4B 0   INZ(0)
DBYTESOUT    5      8B 0   INZ(0)
C*
C*****
C* START OF PROGRAM *
C* * * * *
C   *ENTRY      PLIST
C             PARM                FILEPARM      50
C*****
C* Open certificate file
C*****
C* *-----*
C* ** Build path name *
C* *-----*
C             EVAL      PATHLEN = %LEN(%TRIM(FILEPARM))
C   PATHLEN     SUBST    FILEPARM:1  PATH
C* *-----*
C* * Open the file *
C* *-----*
C             EVAL      FILED = open(PATH: OFLAG)
C* *-----*
C* * Check if open worked *
C* *-----*
C   FILED       IFEQ     -1
C* *-----*
C* * Open failed, send an error message *
C* *-----*
C             MOVEL     MSG(1)      MSGTEXT
C             EXSR      SNDMSG
C             RETURN
C*
C             ENDIF
C* *-----*
C* * Open worked, read certificate and close the file *
C* *-----*
C             EVAL      TOKENLEN = read(FILED: TOKEN: TOKENLEN)
C             CALLP     close      (FILED)
C*
C* *-----*
C* * Check if read operation was OK *
C* *-----*
C   TOKENLEN    IFEQ     -1
C             MOVEL     MSG(2)      MSGTEXT
C             EXSR      SNDMSG
C             RETURN
C             ENDIF
C*
C* *-----*
C* * Check if certificate length is valid *
C* * The length bytes start at position 3 *
C* *-----*
C             EVAL      MSB = TOKENARRAY(3)
C             EVAL      LSB = TOKENARRAY(4)
C   LENGTH      IFLT     TOKENLEN
C* *-----*

```

```

C*      * Certificate length is not valid *
C*      *-----*
C          MOVEL      MSG(3)      MSGTEXT
C          EXSR      SNDMSG
C          RETURN
C          ENDIF
C*
C*****
C* Find the certificate in the token
C*
C* The layout of the token is
C*
C* - Token header - 8 bytes - including 2 length bytes
C* - Public key section - length bytes at position 3 (11 overall)
C* - Private key name - 68 bytes
C* - Certificate section
C*
C* Note: 1 is added because RPG arrays start at 1.
C*****
C          EVAL      MSB = TOKENARRAY(11)
C          EVAL      LSB = TOKENARRAY(12)
C          EVAL      PUBSECLN = LENGTH
C          EVAL      TKNINDEX = PUBSECLN + 68 + 8 + 1
C*
C*      *-----*
C*      * Determine length of certificate section *
C*      * Length bytes are at position 2 of the *
C*      * section.
C*      *-----*
C          EVAL      MSB = TOKENARRAY(TKNINDEX + 2)
C          EVAL      LSB = TOKENARRAY(TKNINDEX + 3)
C          EVAL      CRTSECLN = LENGTH
C          EVAL      TMPINDEX = TKNINDEX + 4
C*
C*      *-----*
C*      * Parse each subsection of the certificate until the *
C*      * signature subsection is found or the end is reached.*
C*      * (Identifier for signature subsection is Hex 45.) *
C*      *-----*
C          DOW      (TOKENARRAY(TMPINDEX) <> X'45') AND
C                  (TMPINDEX < TKNINDEX + CRTSECLN)
C          EVAL      MSB = TOKENARRAY(TMPINDEX + 2)
C          EVAL      LSB = TOKENARRAY(TMPINDEX + 3)
C          TMPINDEX  ADD      LENGTH      TMPINDEX
C          ENDDO
C*
C*      *-----*
C*      * Check if no signature was found before the end of *
C*      * the certificate section.
C*      *-----*
C          IF      TOKENARRAY(TMPINDEX) <> X'45'
C          MOVEL      MSG(4)      MSGTEXT
C          EXSR      SNDMSG
C          RETURN
C          ENDIF
C*
C*****
C* Hash the certificate
C*****
C*      *-----*
C*      * Calculate the length to hash *
C*      *-----*
C          EVAL      TXTLENGTH = TMPINDEX - TKNINDEX + 70
C*      *-----*
C*      * Set the keywords in the rule array *
C*      *-----*
C          MOVEL      'SHA-1 '      RULEARRAY

```

```

C          Z-ADD      1          RULEARRAYCNT
C*  *-----*
C*  * Call One Way Hash SAPI *
C*  *-----*
C          CALLP      CSNBOWH    (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     TXTLENGTH:
C                                     TOKENARRAY(TKNINDEX):
C                                     CHAINVCTLEN:
C                                     CHAINVCT:
C                                     HASHLEN:
C                                     HASH)
C*  *-----*
C*  * Check the return code *
C*  *-----*
C          RETURNCODE  IFGT      0
C*  *-----*
C*  * Send failure message *
C*  *-----*
C          MOVE      MSG(5)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC
C          MOVE      REASONCODE  FAILRSNC
C          MOVE      'CSNBOWH'   SAPI
C          EXSR      SNDMSG
C          RETURN
C          ENDIF
C*
C*****
C* Register the certificate hash
C*****
C*  *-----*
C*  * Set the keywords in the rule array *
C*  *-----*
C          MOVE      'SHA-1 '    RULEARRAY
C          MOVE      'CLONE '    RULEARRAY
C          Z-ADD      2          RULEARRAYCNT
C*  *-----*
C*  * Build the key name (FILENAME.RETAINED) *
C*  *-----*
C          EVAL      %SUBST(NAME: 1: PATHLEN) =
C                                     %SUBST(PATH: 1: PATHLEN)
C          EVAL      %SUBST(NAME:PATHLEN+1:9) = '.RETAINED'
C*  *-----*
C*  * Call PKA Public Key Hash Register *
C*  *-----*
C          CALLP      CSNDPKH    (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     NAME:
C                                     HASHLEN:
C                                     HASH)
C*  *-----*
C*  * Check the return code *
C*  *-----*
C          RETURNCODE  IFGT      0
C*  *-----*
C*  * Send failure message *
C*  *-----*
C          MOVE      MSG(5)      MSGTEXT

```

```

|      C          MOVE      RETURNCODE      FAILRETC
|      C          MOVE      REASONCODE      FAILRSNC
|      C          MOVE      'CSNDPKH'      SAPI
|      C          EXSR      SNDMSG
|      C          ELSE
|      C*         *-----*
|      C*         *  Send success message *
|      C*         *-----*
|      C          MOVE      MSG(6)          MSGTEXT
|      C          EVAL      %SUBST(MSGTEXT: 41: PATHLEN + 9) =
|      C                               %SUBST(NAME: 1: PATHLEN + 9)
|      C          EXSR      SNDMSG
|      C          ENDIF
|      C*
|      C          SETON                                     LR
|      C*
|      C*****
|      C* Subroutine to send a message
|      C*****
|      C          SNDMSG      BEGSR
|      C          CALL      'QMHSNDPM'
|      C          PARM          MESSAGEID
|      C          PARM          MESSAGEFILE
|      C          PARM          MSGTEXT
|      C          PARM          MSGLENGTH
|      C          PARM          MSGTYPE
|      C          PARM          STACKENTRY
|      C          PARM          STACKCOUNTER
|      C          PARM          MSGKEY
|      C          PARM          ERRCODE
|      C          ENDSR
|
| **
| The file could not be opened.
| There was an error reading from the file.
| The length of the certificate is not valid.
| The certificate is not valid.
| CSNBOWH failed with return/reason codes 9999/9999.
| The hash was successfully registered as

```

例: 公開鍵証明書を登録するための ILE C プログラム:

公開鍵の証明書を登録するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| /*-----*/
| /* REGPUBKEY */
| /*
| /* Sample program to register a CCA public key certificate */
| /*
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 1999 */
| /*
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /*
| /*
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */

```

```

| /*      (SC31-8609) publication.          */
| /*                                          */
| /* Parameters: Stream file containing public key certificate */
| /*                                          */
| /* Example:                               */
| /*   CALL PGM(REGPUBKEY) PARM(CERTFILE)   */
| /*                                          */
| /*                                          */
| /* Note: This program assumes the card with the profile is   */
| /*       already identified either by defaulting to the CRP01 */
| /*       device or by being explicitly named using the       */
| /*       Cryptographic_Resource_Allocate verb. Also this    */
| /*       device must be varied on and you must be authorized */
| /*       to use this device description.     */
| /*                                          */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* PKA_Public_Key_Register (CSNDPKR).     */
| /*                                          */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA)                                           */
| /* CRTCMOD MODULE(REGPUBKEY) SRCFILE(SAMPLE)                 */
| /* CRTPGM  PGM(REGPUBKEY)  MODULE(REGPUBKEY)                 */
| /*          BNDDIR(QCCA/QC6BNDDIR)                           */
| /*                                          */
| /* Note: Authority to the CSNDPKR service program            */
| /*       in the QCCA library is assumed.                     */
| /*                                          */
| /*-----*/
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h"
|
| int main(int argc, char *argv[])
| {
| /*-----*/
| /* Declares for CCA parameters                               */
| /*-----*/
| long return_code = 0;
| long reason_code = 0;
| long exit_data_length = 0;
| char exit_data[4];
| char rule_array[24];
| long rule_array_count;
| long token_len = 2500;
| char token[2500];
| /*-----*/
| /* Declares for working with a PKA token                    */
| /*-----*/
| long pub_sec_len;          /* Public section length      */
| long cert_sec_len;        /* Certificate section length */
| long offset;              /* Offset into token         */
| long tempOffset;          /* (Another) Offset into token */
| char name[64];            /* Registered key name       */
|
| long count;               /* Number of bytes read from file */
| FILE *fp;                 /* File pointer               */
|
| if (argc < 2)             /* Check the number of parameters passed */
| {
|     printf("Need to enter a public key name\n");
|     return 1;
| }
|
| memset(name, ' ',64);     /* Copy key name (and pad) to a 64 byte */
|                          /* field.                               */
| memcpy(name,argv[1],strlen(argv[1]));

```

```

| fp = fopen(argv[1],"rb"); /* Open the file for reading */
| if (!fp)
| {
|     printf("File %s not found.\n",argv[1]);
|     return 1;
| }
|
| memset(token,0,2500); /* Initialize the token to 0 */
| count = fread(token,1,2500,fp); /* Read the token from the file */
| fclose(fp); /* Close the file */
|
| /* Determine length of token from length */
| /* bytes at offset 2 and 3. */
| token_len = ((256 * token[2]) + token[3]);
| if (count < token_len) /* Check if whole token was read in */
| {
|     printf("Incomplete token in file\n");
|     return 1;
| }
|
| /*****
| /* Find the certificate length in the token */
| /*
| /* The layout of the token is
| /*
| /* - Token header - 8 bytes - including 2 length bytes */
| /* - Public key section - length bytes at offset 2 */
| /* - Private key name - 68 bytes */
| /* - Certificate section */
| *****/
| pub_sec_len = ((256 * token[10]) + token[11]);
|
| offset = pub_sec_len + 68 + 8; /* Set offset to certificate section */
|
| /* Determine certificate section */
| /* length from the length bytes at */
| /* offset 2 of the section. */
| cert_sec_len = ((256 * token[offset + 2]) + token[offset + 3]);
|
| /*****
| /* Register the Public Key */
| *****/
| memcpy((void*)rule_array,"CLONE ",8); /* Set rule array */
| rule_array_count = 1;
|
| /* Build the name of the retained */
| /* key from the file and "RETAINED"*/
| memcpy(&name[strlen(argv[1])],".RETAINED",9);
|
| CSNDPKR( &return_code, &reason_code, &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char*)rule_array,
|         name,
|         &cert_sec_len,
|         &token[offset]);
|
| if (return_code != 0)
| {
|     printf("Public Key Register Failed : return reason %d/%d\n",
|           return_code, reason_code);
|     return 1;
| }
|
| name[strlen(argv[1]) + 9] = 0; /* Convert name to a string */
| printf("Public key registered for %s.\n",name);
|
| }

```

例: 公開鍵証明書を登録するための ILE RPG プログラム:

公開鍵の証明書を登録するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```
D*****
D* REGPUBKEY
D*
D* Sample program to register a CCA public key
D* certificate.
D*
D*
D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
D*
D* This material contains programming source code for your
D* consideration. These example has not been thoroughly
D* tested under all conditions. IBM, therefore, cannot
D* guarantee or imply reliability, serviceability, or function
D* of these programs. All programs contained herein are
D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
D* these programs and files.
D*
D*
D* Note: Input format is more fully described in Chapter 2 of
D*       IBM CCA Basic Services Reference and Guide
D*       (SC31-8609) publication.
D*
D* Parameters: Stream file containing public key certificate
D*
D* Example:
D*   CALL PGM(REGPUBKEY) PARM(CERTFILE)
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(REGPUBKEY) SRCFILE(SAMPLE)
D* CRTPGM PGM(REGPUBKEY) MODULE(REGPUBKEY)
D*       BNDDIR(QCCA/QC6BNDDIR)
D*
D* Note: Authority to the CSNDPKR service program
D*       in the QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* PKA_Public_Key_Register (CSNDPKR).
D*
D*****
D*-----
D* Declare variables used by CCA SAPI calls
D*-----
D*           ** Return code
DRETURNCODE S           9B 0
D*           ** Reason code
DREASONCODE S           9B 0
D*           ** Exit data length
DEXITDATALEN S          9B 0
D*           ** Exit data
DEXITDATA S             4
D*           ** Rule array count
DRULEARRAYCNT S         9B 0
D*           ** Rule array
DRULEARRAY S           16
D*           ** Token length
DTOKENLEN S            9B 0 INZ(2500)
D*           ** Token and array for subscripting token
```



```

| DTOKEN          DS          2500
| DTOKENARRAY    S          1    DIM(2500)
| D*              ** Name of retained key
| DNAME          S          64
| D*              ** Structure used for aligning 2 bytes into a
| D*              ** 2 byte integer.
| DLENSTRUCT     DS          2
| DMSB           S          1    1
| DLSB           S          2    2
| DLENGTH        S          1    2B 0
| D*              ** Certificate section length
| DCRTSECLN     S          9B 0
| D*              ** Public key section length
| DPUBSECLN     S          9B 0
| D*              ** Index into PKA key token
| DTKNINDEX     S          9B 0
| D*              ** Index into PKA key token
| DTMPINDEX     S          9B 0
| D*              ** File descriptor
| DFILED        S          9B 0
| D*              ** File path and path length
| DPATH         S          80    INZ(*ALLX'00')
| DPATHLEN      S          9B 0
| D*              ** Open Flag - Open for Read only
| DOFLAG        S          10I 0 INZ(1)
| D*
| D*****
| D* Prototype for PKA_Public_Key_Register (CSNDPKR)
| D*****
| DCSNDPKR      PR
| DRETCOD       S          9B 0
| DRSNCOD       S          9B 0
| DEXTDTALN     S          9B 0
| DEXTDT        S          4
| DRARRYCT      S          9B 0
| DRARRY        S          16
| DKYNAM        S          64
| DCRTLEN       S          9B 0
| DCRT          S          500   OPTIONS(*VARSIZE)
| D*
| D*****
| D* Prototype for open()
| D*****
| D* value returned = file descriptor (OK), -1 (error)
| Dopen         PR          9B 0 EXTPROC('open')
| D* path name of file to be opened.
| D            S          128   OPTIONS(*VARSIZE)
| D* Open flags
| D            S          9B 0 VALUE
| D* (OPTIONAL) mode - access rights
| D            S          10U 0 VALUE OPTIONS(*NOPASS)
| D* (OPTIONAL) codepage
| D            S          10U 0 VALUE OPTIONS(*NOPASS)
| D*
| D*****
| D* Prototype for read()
| D*****
| D* value returned = number of bytes actually read, or -1
| Dread        PR          9B 0 EXTPROC('read')
| D* File descriptor returned from open()
| D            S          9B 0 VALUE
| D* Input buffer
| D            S          2500  OPTIONS(*VARSIZE)
| D* Length of data to be read
| D            S          9B 0 VALUE
| D*
| D*****

```

```

D* Prototype for close()
D*****
D*   value returned = 0 (OK), or -1
Dclose      PR          9B 0 EXTPROC('close')
D*   File descriptor returned from open()
D           9B 0 VALUE
D*
D*-----
D*           ** Declares for sending messages to the
D*           ** job log using the QMHSNDPM API
D*-----
DMSG        S          75   DIM(5) CTDATA PERRCD(1)
DMSGLENGTH S          9B 0 INZ(75)
D           DS
DMSGTEXT    1          80
DFAILRETC  41         44
DFAILRSNC  46         49
DMESSAGEID S          7   INZ('      ')
DMESSAGEFILE S        21  INZ('          ')
DMSGKEY     S          4   INZ('      ')
DMSGTYPE    S         10  INZ('*INFO  ')
DSTACKENTRY S         10  INZ('*      ')
DSTACKCOUNTER S       9B 0 INZ(2)
DERRCODE    DS
DBYTESIN    1         4B 0 INZ(0)
DBYTESOUT   5         8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM
C*
C   *ENTRY      PLIST
C               PARM          FILEPARM      50
C*****
C* Open certificate file
C*****
C* *-----*
C* ** Build path name *
C* *-----*
C           EVAL      PATHLEN = %LEN(%TRIM(FILEPARM))
C   PATHLEN     SUBST  FILEPARM:1  PATH
C* *-----*
C* * Open the file *
C* *-----*
C           EVAL      FILED = open(PATH: OFLAG)
C* *-----*
C* * Check if open worked *
C* *-----*
C   FILED       IFEQ    -1
C* *-----*
C* * Open failed, send an error message *
C* *-----*
C           MOVEL     MSG(1)      MSGTEXT
C           EXSR      SNDMSG
C           RETURN
C*
C           ENDIF
C* *-----*
C* * Open worked, read certificate and close the file *
C* *-----*
C           EVAL      TOKENLEN = read(FILED: TOKEN: TOKENLEN)
C           CALLP     close      (FILED)
C*
C* *-----*
C* * Check if read operation was OK *
C* *-----*
C   TOKENLEN    IFEQ    -1
C           MOVEL     MSG(2)      MSGTEXT

```

```

C          EXSR      SNDMSG
C          RETURN
C          ENDIF
C*
C* -----*
C* * Check if certificate length is valid *
C* * The length bytes start at position 3 *
C* -----*
C          EVAL      MSB = TOKENARRAY(3)
C          EVAL      LSB = TOKENARRAY(4)
C LENGTH          IFLT      TOKENLEN
C* -----*
C* * Certificate length is not valid *
C* -----*
C          MOVE      MSG(3)      MSGTEXT
C          EXSR      SNDMSG
C          RETURN
C          ENDIF
C*
C*****
C* Find the certificate in the token
C*
C* The layout of the token is
C*
C* - Token header - 8 bytes - including 2 length bytes
C* - Public key section - length bytes at position 3 (11 overall)
C* - Private key name - 68 bytes
C* - Certificate section
C*
C* Note: 1 is added because RPG arrays start at 1.
C*****
C          EVAL      MSB = TOKENARRAY(11)
C          EVAL      LSB = TOKENARRAY(12)
C          EVAL      PUBSECLN = LENGTH
C          EVAL      TKNINDEX = PUBSECLN + 68 + 8 + 1
C*
C* -----*
C* * Determine length of certificate section *
C* * Length bytes are at position 2 of the *
C* * section.
C* -----*
C          EVAL      MSB = TOKENARRAY(TKNINDEX + 2)
C          EVAL      LSB = TOKENARRAY(TKNINDEX + 3)
C          EVAL      CRTSECLN = LENGTH
C*
C*****
C* Register the public key
C*****
C* -----*
C* * Set the keywords in the rule array *
C* -----*
C          MOVE      'CLONE '      RULEARRAY
C          Z-ADD      1              RULEARRAYCNT
C* -----*
C* * Build the key name (FILENAME.RETAINED) *
C* -----*
C          EVAL      %SUBST(NAME: 1: PATHLEN) =
C                  %SUBST(PATH: 1: PATHLEN)
C          EVAL      %SUBST(NAME:PATHLEN+1:9) = '.RETAINED'
C* -----*
C* * Call PKA Public Key Register *
C* -----*
C          CALLP      CSNDPKR      (RETURNCODE:
C                                  REASONCODE:
C                                  EXITDATALEN:
C                                  EXITDATA:

```

```

|      C                                RULEARRAYCNT:
|      C                                RULEARRAY:
|      C                                NAME:
|      C                                CRTSECLN:
|      C                                TOKENARRAY(TKNINDEX))
|
|      C* *-----*
|      C* * Check the return code *
|      C* *-----*
|      C      RETURNCODE      IFGT      0
|      C* *-----*
|      C* * Send failure message *
|      C* *-----*
|      C                                MOVE      MSG(4)      MSGTEXT
|      C                                MOVE      RETURNCODE  FAILRETC
|      C                                MOVE      REASONCODE  FAILRSNC
|      C                                EXSR      SNDMSG
|      C                                ELSE
|      C* *-----*
|      C* * Send success message *
|      C* *-----*
|      C                                MOVE      MSG(5)      MSGTEXT
|      C                                EVAL      %SUBST(MSGTEXT: 41: PATHLEN + 9) =
|      C                                %SUBST(NAME: 1: PATHLEN + 9)
|      C                                EXSR      SNDMSG
|      C                                ENDIF
|      C*
|      C                                SETON
|
|      C*
|      C*-----*
|      C* Subroutine to send a message
|      C*-----*
|      C      SNDMSG      BEGSR
|      C                                CALL      'QMHSNDPM'
|      C                                PARM
|      C                                PARM      MESSAGEID
|      C                                PARM      MESSAGEFILE
|      C                                PARM      MSGTEXT
|      C                                PARM      MSGLENGTH
|      C                                PARM      MSGTYPE
|      C                                PARM      STACKENTRY
|      C                                PARM      STACKCOUNTER
|      C                                PARM      MSGKEY
|      C                                PARM      ERRCODE
|      C                                ENDSR
|
|      **
|      The file could not be opened.
|      There was an error reading from the file.
|      The length of the certificate is not valid.
|      CSNDPKR failed with return/reason codes 9999/9999.
|      The hash was successfully registered as

```

例: 公開鍵のトークンを認証するための ILE C プログラム:

公開鍵のトークンを認証するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| /*-----*/
| /* CERTKEY */
| /*
| /* Sample program to certify a CCA public key certificate to be
| /* used for master key cloning.
| /*
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 1999
| /*
| /* This material contains programming source code for your

```

```

| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: FILENAME - File containing public key token */
| /* RETAINED_KEY_NAME - Name of key to certify token */
| /* */
| /* Example: */
| /* CALL PGM(CERTKEY) PARM(MYKEY.PUB CERTKEY) */
| /* */
| /* Note: This program assumes the card with the profile is */
| /* already identified either by defaulting to the CRP01 */
| /* device or by being explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verbs used are */
| /* Digital_Signature_Generate (CSNDDSG) and One_Way_Hash (CSNBOWH). */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCPGM MODULE(CERTKEY) SRCFILE(SAMPLE) */
| /* CRTCPGM PGM(CERTKEY) MODULE(CERTKEY) */
| /* BNDDIR(QCCA/QC6BNDDIR) */
| /* */
| /* Note: Authority to the CSNDDSG and CSNBOWH service programs */
| /* in the QCCA library is assumed. */
| /* */
| /*-----*/
| #include <stdio.h>
| #include <string.h>
| #include "csucincl.h"
| #include "decimal.h"
|
| extern void QDCXLATE(decimal(5,0), char *, char*, char *);
| #pragma linkage (QDCXLATE, OS, nowiden)
|
| int main(int argc, char *argv[])
| {
| /*-----*/
| /* Declares for CCA parameters */
| /*-----*/
| long return_code = 0;
| long reason_code = 0;
| long exit_data_length = 0;
| char exit_data[4];
| char rule_array[24];
| long rule_array_count;
| long token_len = 2500;
| char token[2500];
| long chaining_vector_length = 128;
| long hash_length = 20;
| long text_length;
| unsigned char chaining_vector[128];
| unsigned char hash[20];

```

```

| long signature_length = 256;
| long signature_bit_length;
| /*-----*/
| /* Declares for working with a PKA token */
| /*-----*/
| long pub_sec_len; /* Public section length */
| long cert_sec_len; /* Certificate section length */
| long offset; /* Offset into token */
| long tempOffset; /* (Another) Offset into token */
| long tempLength; /* Length variable */
| char name[64]; /* Private key name */
| char SName[64]; /* Share administration or certifying
| /* key name.
| char SNameASCII[64]; /* Share admin key name in ASCII
| long SName_length = 64; /* Length of Share admin key name
| long count; /* Number of bytes read from file
| decimal(5,0) xlate_length = 64; /* Packed decimal variable
| /* needed for call to QDCXLATE.
| FILE *fp; /* File pointer
|
| if (argc < 3) /* Check the number of parameters passed */
| {
| printf("Need to enter a public key name and SA key\n");
| return 1;
| }
|
| name[0] = 0; /* Make copy of name parameters */
| strcpy(name,argv[1]);
| memset(SName, ' ', 64); /* Make copy of Share Admin key name
| memcpy(SName,argv[2],strlen(argv[2]));
|
| fp = fopen(name,"rb"); /* Open the file containing the token */
| if (!fp)
| {
| printf("File %s not found.\n",argv[1]);
| return 1;
| }
|
| memset(token,0,2500); /* Read the token from the file */
| count = fread(token,1,2500,fp);
| fclose(fp);
|
| /* Determine length of token from length */
| /* bytes at offset 2 and 3. */
| token_len = ((256 * token[2]) + token[3]);
| if (count < token_len) /* Check if whole token was read in */
| {
| printf("Incomplete token in file\n");
| return 1;
| }
|
| /*-----*/
| /* Find the certificate offset in the token */
| /*
| /* The layout of the token is
| /*
| /* - Token header - 8 bytes - including 2 length bytes
| /* - Public key section - length bytes at offset 10 overall
| /* - Private key name - 68 bytes
| /* - Certificate section
| /*
| /*
| /*-----*/
| pub_sec_len = ((256 * token[10]) + token[11]);
|
| offset = pub_sec_len + 68 + 8; /* Set offset to certiiate section */
|
| /* Determine certificate section */
| /* length from the length bytes at

```

```

|                                     /* offset 2 of the section.      */
| cert_sec_len = ((256 * token[offset + 2]) + token[offset + 3]);
| tempOffset = offset + 4;          /* Set offset to first subsection */
|
|                                     /*-----*/
|                                     /* Parse each subsection of the certificate until the */
|                                     /* signature subsection is found or the end is reached.*/
|                                     /* (Identifier for signature subsection is Hex 45.)  */
|                                     /*-----*/
| while(token[tempOffset] != 0x45 &&
|       tempOffset < offset + cert_sec_len)
| {
|   tempOffset += 256 * token[tempOffset + 2] + token[tempOffset+3];
| }
|
|                                     /*-----*/
|                                     /* Check if no signature was found before the end of */
|                                     /* the certificate section.          */
|                                     /*-----*/
| if (token[tempOffset] != 0x45)
| {
|   printf("Invalid certificate\n");
|   return 1;
| }
|
|                                     /*-----*/
|                                     /* Replace Private key name in certificate with the */
|                                     /* Share admin key name (expressed in ASCII). */
|                                     /*-----*/
| text_length = tempOffset - offset + 70;
| memcpy(SAnameASCII,SAname,64);
|                                     /*-----*/
|                                     /* Convert the Share Admin key name to ASCII */
|                                     /*-----*/
| QDCXLATE(xlate_length, SAnameASCII, "QASCII ", "QSYS ");
| memcpy(&token[tempOffset + 6], SAnameASCII, 64);
|
|                                     /*-----*/
|                                     /* Hash the certificate */
|                                     /*-----*/
| memcpy((void*)rule_array,"SHA-1 ",8);
| rule_array_count = 1;
| chaining_vector_length = 128;
| hash_length = 20;
|
| CSNBOWH( &return_code, &reason_code, &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char*)rule_array,
|         &text_length,
|         &token[offset],
|         &chaining_vector_length,
|         chaining_vector,
|         &hash_length,
|         hash);
|
| if (return_code != 0)
| {
|   printf("One_Way_Hash Failed : return reason %d/%d\n",
|         return_code, reason_code);
|   return 1;
| }
|
|                                     /*-----*/
|                                     /* Create a signature */
|                                     /*-----*/
| memcpy((void*)rule_array,"ISO-9796",8);

```

```

rule_array_count = 1;

CSNDDSG( &return_code, &reason_code, &exit_data_length,
        exit_data,
        &rule_array_count,
        (unsigned char*)rule_array,
        &SAname_length,
        SAname,
        &hash_length,
        hash,
        &signature_length,
        &signature_bit_length,
        &token[tempOffset+70]);

if (return_code != 0)
{
    printf("Digital Signature Generate Failed : return reason %d/%d\n",
        return_code, reason_code);
    return 1;
}

/*-----*/
/* Check if the new signature is longer than the */
/* original signature */
/*-----*/
if((token[tempOffset + 2] * 256 + token[tempOffset + 3]) - 70 !=
    signature_length)
{
    printf("Signature Length change from %d to %d.\n",
        token[tempOffset + 2] * 256 + token[tempOffset + 3] - 70,
        signature_length);

    /* Adjust length in signature subsection */
    token[tempOffset + 2] = signature_length >> 8;
    token[tempOffset + 3] = signature_length;

    /* Adjust length in certificate section */
    token[offset + 2] = (text_length + signature_length) >> 8;
    token[offset + 3] = text_length + signature_length;

    /* Adjust length in token header section */
    tempLength = 8 + pub_sec_len + 68 + text_length +
        signature_length;
    token[2] = tempLength >> 8;
    token[3] = tempLength;
}
else tempLength = token[2] * 256 + token[3];

/*****
/* Write certified public key out to a file */
*****/
strcat(name, ".CRT"); /* Append .CRP to filename */
fp = fopen(name, "wb"); /* Open the certificate file */
if (!fp)
{
    printf("File open failed for output\n");
}
else
{
    fwrite(token, 1, tempLength, fp);
    fclose(fp);
    printf("Public token written to file %s.\n", name);
}
}

```


例: 公開鍵のトークンを認証するための ILE RPG プログラム:

公開鍵のトークンを認証するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```
D*****
D* CERTKEY
D*
D* Sample program to certify a CCA public key certificate to be
D* used for master key cloning.
D*
D*
D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
D*
D* This material contains programming source code for your
D* consideration. These example has not been thoroughly
D* tested under all conditions. IBM, therefore, cannot
D* guarantee or imply reliability, serviceability, or function
D* of these programs. All programs contained herein are
D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
D* these programs and files.
D*
D*
D* Note: Input format is more fully described in Chapter 2 of
D*       IBM CCA Basic Services Reference and Guide
D*       (SC31-8609) publication.
D*
D* Parameters: FILENAME          - File containing public key token
D*              RETAINED_KEY_NAME - Name of key to certify token
D*
D* Example:
D* CALL PGM(CERTKEY) PARM(MYKEY.PUB CERTKEY)
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(CERTKEY) SRCFILE(SAMPLE)
D* CRTPGM PGM(CERTKEY) MODULE(CERTKEY)
D*       BNDDIR(QCCA/QC6BNDDIR)
D*
D* Note: Authority to the CSNDDSG and CSNBOWH service programs
D*       in the QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Digital_Signature_Generate (CSNDDSG) and One_Way_Hash (CSNBOWH).
D*
D*****
D*-----
D* Declare variables used by CCA SAPI calls
D*-----
D*          ** Return code
DRETURNCODE S          9B 0
D*          ** Reason code
DREASONCODE S          9B 0
D*          ** Exit data length
DEXITDATALEN S         9B 0
D*          ** Exit data
DEXITDATA S           4
D*          ** Rule array count
DRULEARRAYCNT S        9B 0
D*          ** Rule array
DRULEARRAY S           16
D*          ** Token length
DTOKENLEN S           9B 0 INZ(2500)
```

```

|      D*          ** Token and array for subscribing token
|      DTOKEN      DS          2500
|      DTOKENARRAY          1    DIM(2500)
|      D*          ** Chaining vector length
|      DCHAINVCTLEN S          9B 0 INZ(128)
|      D*          ** Chaining vector
|      DCHAINVCT   S          128
|      D*          ** Hash length
|      DHASHLEN    S          9B 0 INZ(20)
|      D*          ** Hash
|      DHASH       S          20
|      D*          ** Text length
|      DTXTLENGTH S          9B 0
|      D*          ** Signature length
|      DSIGLENGTH S          9B 0 INZ(256)
|      D*          ** Signature length in bits
|      DSIGBITLEN S          9B 0
|      D*-----
|      D* Declare variables for working with tokens
|      D*-----
|      D*          ** NAMEPTR and NAME are used for copying
|      D*          ** private key name
|      DNAMEPTR    S          *
|      DNAME       S          64   BASED(NAMEPTR)
|      D*          ** Share administrator (certifying key) name length
|      DSANAMELEN S          9B 0
|      D*          ** Share administrator (certifying key) name
|      DSANAME     S          64
|      D*          ** Share administrator name expressed in ASCII
|      DSANAMEASC S          64
|      D*          ** Certificate section length
|      DCRTSECLN  S          9B 0
|      D*          ** Public key section length
|      DPUBSECLN  S          9B 0
|      D*          ** Index into PKA key token
|      DTKNINDEX  S          9B 0
|      D*          ** Index into PKA key token
|      DTMPINDEX  S          9B 0
|      D*          ** Structure used for aligning 2 bytes into a
|      D*          ** 2 byte integer.
|      DLENSTRUCT DS          2
|      DMSB       1          1
|      DLSB       2          2
|      DLENGTH    1          2B 0
|      D*          ** File descriptor
|      DFILED     S          9B 0
|      D*          ** File path and path length
|      DPATH      S          80   INZ(*ALLX'00')
|      DPATHLEN   S          9B 0
|      D*          ** Open flag - Create on open, open for writing,
|      D*          ** and clear if exists
|      DOFLAGW    S          10I 0 INZ(X'4A')
|      D*          ** Open Flag - Open for Read only
|      DOFLAGR    S          10I 0 INZ(1)
|      D*          ** Declares for calling QDCXLATE API
|      DXTABLE    S          10   INZ('QASCII  ')
|      DLIB       S          10   INZ('QSYS    ')
|      DXLATLEN   S          5   0 INZ(64)
|      D
|      D*
|      D*****
|      D* Prototype for Digital_Signature_Generate (CSNDDSG)
|      D*****
|      DCSNDDSG   PR
|      DRETCOD    9B 0
|      DRSNCOD    9B 0
|      DEXTDTALN  9B 0

```

```

| DEXTDT 4
| DRARRYCT 9B 0
| DRARRY 16
| DKEYIDLEN 9B 0
| DKEYID 2500 OPTIONS(*VARSIZE)
| DHSHL 9B 0
| DSH 20 OPTIONS(*VARSIZE)
| DSIGFLDL 9B 0
| DSIGBTL 9B 0
| DSIGFLD 256 OPTIONS(*VARSIZE)
| D*
| D*****
| D* Prototype for One_Way_Hash (CSNBOWH)
| D*****
| DCSNBOWH PR
| DRETCOD 9B 0
| DRNSCOD 9B 0
| DEXTDTALN 9B 0
| DEXTDT 4
| DRARRYCT 9B 0
| DRARRY 16
| DTXTLEN 9B 0
| DTXT 500 OPTIONS(*VARSIZE)
| DCHNVCTLEN 9B 0
| DCHNVCT 128
| DSHLEN 9B 0
| DSH 20
| D*
| D*
| D*****
| D* Prototype for open()
| D*****
| D* value returned = file descriptor (OK), -1 (error)
| Dopen PR 9B 0 EXTPROC('open')
| D* path name of file to be opened.
| D 128 OPTIONS(*VARSIZE)
| D* Open flags
| D 9B 0 VALUE
| D* (OPTIONAL) mode - access rights
| D 10U 0 VALUE OPTIONS(*NOPASS)
| D* (OPTIONAL) codepage
| D 10U 0 VALUE OPTIONS(*NOPASS)
| D*
| D*****
| D* Prototype for read()
| D*****
| D* value returned = number of bytes actually read, or -1
| Dread PR 9B 0 EXTPROC('read')
| D* File descriptor returned from open()
| D 9B 0 VALUE
| D* Input buffer
| D 2500 OPTIONS(*VARSIZE)
| D* Length of data to be read
| D 9B 0 VALUE
| D*
| D*****
| D* Prototype for write()
| D*****
| D* value returned = number of bytes written, or -1
| Dwrite PR 9B 0 EXTPROC('write')
| D* File descriptor returned from open()
| D 9B 0 VALUE
| D* Output buffer
| D 2500 OPTIONS(*VARSIZE)
| D* Length of data to be written
| D 9B 0 VALUE
| D*

```

```

D*****
D* Prototype for close()
D*****
D*   value returned = 0 (OK), or -1
Dclose      PR          9B 0 EXTPROC('close')
D*   File descriptor returned from open()
D           9B 0 VALUE
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG        S          75   DIM(7) CTDATA PERRCD(1)
DMSGLENGTH S          9B 0 INZ(75)
D           DS
DMSGTEXT    1          75
DSAPI       1          7
DFAILRETC  41         44
DFAILRSNC   46         49
DMESSAGEID  S          7   INZ(' ')
DMESSAGEFILE S        21   INZ(' ')
DMSGKEY     S          4   INZ(' ')
DMSGTYPE    S          10  INZ('*INFO ')
DSTACKENTRY S          10  INZ('* ')
DSTACKCOUNTER S        9B 0 INZ(2)
DERRCODE    DS
DBYTESIN    1          4B 0 INZ(0)
DBYTESOUT   5          8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM *
C*****
C   *ENTRY      PLIST
C               PARM          FILEPARM      32
C               PARM          CKEY          32
C*****
C* Open certificate file
C*****
C* *-----*
C* ** Build path name *
C* *-----*
C               EVAL          PATHLEN = %LEN(%TRIM(FILEPARM))
C   PATHLEN     SUBST          FILEPARM:1   PATH
C* *-----*
C* * Open the file *
C* *-----*
C               EVAL          FILED = open(PATH: OFLAGR)
C* *-----*
C* * Check if open worked *
C* *-----*
C   FILED       IFEQ          -1
C* *-----*
C* * Open failed, send an error message *
C* *-----*
C               MOVEL          MSG(1)      MSGTEXT
C               EXSR          SNDMSG
C               RETURN
C*
C               ENDIF
C* *-----*
C* * Open worked, read certificate and close the file *
C* *-----*
C               EVAL          TOKENLEN = read(FILED: TOKEN: TOKENLEN)
C               CALLP          close      (FILED)
C*
C* *-----*
C* * Check if read operation was OK *

```

```

C* -----*
C  TOKENLEN      IFEQ      -1
C                MOVEL     MSG(2)      MSGTEXT
C                EXSR      SNDMSG
C                ENDIF
C*
C* -----*
C* * Check if certificate length is valid *
C* -----*
C                EVAL      MSB = TOKENARRAY(3)
C                EVAL      LSB = TOKENARRAY(4)
C  LENGTH        IFLT      TOKENLEN
C* -----*
C* * Certificate length is not valid *
C* -----*
C                MOVEL     MSG(3)      MSGTEXT
C                EXSR      SNDMSG
C                RETURN
C                ENDIF
C*
C*****
C* Find the certificate in the token
C*
C* The layout of the token is
C*
C* - Token header - 8 bytes - including 2 length bytes
C* - Public key section - length bytes at offset 2
C* - Private key name - 68 bytes
C* - Certificate section
C*
C*****
C* -----*
C* * Certificate starts after the public key header section *
C* -----*
C                EVAL      MSB = TOKENARRAY(11)
C                EVAL      LSB = TOKENARRAY(12)
C                EVAL      PUBSECLN = LENGTH
C                EVAL      TKNINDEX = PUBSECLN + 68 + 8 + 1
C*
C* -----*
C* * Determine length of certificate section *
C* -----*
C                EVAL      MSB = TOKENARRAY(TKNINDEX + 2)
C                EVAL      LSB = TOKENARRAY(TKNINDEX + 3)
C                EVAL      CRTSECLN = LENGTH
C                EVAL      TMPINDEX = TKNINDEX + 4
C*
C* -----*
C* * Parse each subsection of the certificate until the *
C* * signature subsection is found or the end is reached.*
C* * (Identifier for signature subsection is Hex 45.) *
C* -----*
C                DOW      (TOKENARRAY(TMPINDEX) <> X'45') AND
C                (TMPINDEX < TKNINDEX + CRTSECLN)
C                EVAL      MSB = TOKENARRAY(TMPINDEX + 2)
C                EVAL      LSB = TOKENARRAY(TMPINDEX + 3)
C  TMPINDEX      ADD      LENGTH      TMPINDEX
C                ENDDO
C*
C* -----*
C* * Check if no signature was found before the end of *
C* * the certificate section. *
C* -----*
C                IF      TOKENARRAY(TMPINDEX) <> X'45'
C                MOVEL     MSG(4)      MSGTEXT
C                EXSR      SNDMSG
C                RETURN

```

```

C          ENDIF
C*
C*****
C* Sign the Certificate
C*****
C* -----*
C* * Convert the Certifying Keyname to ASCII *
C* -----*
C          EVAL          SANAMELEN = %LEN(%TRIM(CKEY))
C          SANAMELEN     SUBST     CKEY:1          SANAME
C          MOVE          SANAME     SANAMEASC
C          CALL          'QDCXLATE'
C          PARM
C          PARM          XLATLEN
C          PARM          SANAMEASC
C          PARM          XTABLE
C          PARM          LIB
C* -----*
C* * Replace the private key name in the certificate *
C* -----*
C          EVAL          NAMEPTR = %ADDR(TOKENARRAY(TMPINDEX + 6))
C          MOVE          SANAMEASC     NAME
C* -----*
C* * Calculate length of data to hash *
C* * TKNINDEX is the start of the certificate, *
C* * TMPINDEX is start of signature subsection, *
C* * signature subsection header is 70 bytes long *
C* -----*
C          EVAL          TXTLENGTH = TMPINDEX - TKNINDEX + 70
C* -----*
C* * Set the keywords in the rule array *
C* -----*
C          MOVE          'SHA-1 '     RULEARRAY
C          Z-ADD         1             RULEARRAYCNT
C* -----*
C* * Call One Way Hash SAPI *
C* -----*
C          CALLP         CSNBOWH      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     TXTLENGTH:
C                                     TOKENARRAY(TKNINDEX):
C                                     CHAINVCTLEN:
C                                     CHAINVCT:
C                                     HASHLEN:
C                                     HASH)
C
C* -----*
C* * Check the return code *
C* -----*
C          RETURNCODE    IFGT         0
C* -----*
C* * Send failure message *
C* -----*
C          MOVE          MSG(5)        MSGTEXT
C          MOVE          RETURNCODE    FAILRETC
C          MOVE          REASONCODE    FAILRSNC
C          MOVE          'CSNBOWH'     SAPI
C          EXSR          SNDMSG
C          RETURN
C          ENDIF
C* -----*
C* * Set the keywords in the rule array *
C* -----*
C          MOVE          'ISO-9796'    RULEARRAY
C          Z-ADD         1             RULEARRAYCNT

```

```

C* *-----*
C* * Adjust TMPINDEX to where signature starts*
C* * in the certificate *
C* *-----*
C   TMPINDEX      ADD      70          TMPINDEX
C* *-----*
C* * Set the Key name length *
C* *-----*
C           Z-ADD      64          SANAMELEN
C* *-----*
C* * Call Digital Signature Generate SAPI *
C* *-----*
C           CALLP      CSNDDSG      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     SANAMELEN:
C                                     SANAME:
C                                     HASHLEN:
C                                     HASH:
C                                     SIGLENGTH:
C                                     SIGBITLEN:
C                                     TOKENARRAY(TMPINDEX))
C* *-----*
C* * Check the return code *
C* *-----*
C   RETURNCODE     IFGT      0
C* *-----*
C* * Send failure message *
C* *-----*
C           MOVE      MSG(5)      MSGTEXT
C           MOVE      RETURNCODE   FAILRETC
C           MOVE      REASONCODE   FAILRSNC
C           MOVE      'CSNDDSG'    SAPI
C           EXSR      SNDMSG
C           RETURN
C           ENDIF
C* *-----*
C* * Check if the new signature is longer than the *
C* * original signature *
C* *-----*
C* ** Adjust TMPINDEX back the start of the subsection
C   TMPINDEX      SUB      70          TMPINDEX
C* ** Get two byte length of subsection
C           EVAL      MSB = TOKENARRAY(TMPINDEX + 2)
C           EVAL      LSB = TOKENARRAY(TMPINDEX + 3)
C* ** Subtract length of subsection header
C   LENGTH      SUB      70          LENGTH
C* ** Compare old length with new length
C   LENGTH      IFNE      SIGLENGTH
C* *-----*
C* * Adjust certificate lengths *
C* *-----*
C* ** Adjust signature length
C           EVAL      LENGTH = SIGLENGTH
C           EVAL      TOKENARRAY(TMPINDEX + 2) = MSB
C           EVAL      TOKENARRAY(TMPINDEX + 3) = LSB
C* ** Adjust certificate section length
C           EVAL      LENGTH = LENGTH + TXTLENGTH
C           EVAL      TOKENARRAY(TKNINDEX + 2) = MSB
C           EVAL      TOKENARRAY(TKNINDEX + 3) = LSB
C* ** Adjust length in token header section
C           EVAL      LENGTH = LENGTH + 8 + PUBSECLN + 68
C           EVAL      TOKENARRAY(3) = MSB

```

```

C          EVAL      TOKENARRAY(4) = LSB
C          Z-ADD     LENGTH      TOKENLEN
C          ENDIF
C*
C*****
C* Write certified public key out to a file
C*****
C*   ** Build path name
C          EVAL      %SUBST(PATH:PATHLEN+1:4) = '.CRT'
C*
C*   ** Open the file
C*
C          EVAL      FILED = open(PATH: OFLAGW)
C*
C*   ** Check if open worked
C*
C          FILED      IFEQ      -1
C*
C*   ** Open failed, send an error message
C*
C          MOVEL     MSG(6)      MSGTEXT
C          EXSR      SNDMSG
C*
C          ELSE
C*
C*   ** Open worked, write certificate out to file and close file
C*
C          CALLP     write      (FILED:
C                                TOKEN:
C                                TOKENLEN)
C          CALLP     close      (FILED)
C*
C*   ** Send completion message
C*
C          MOVEL     MSG(7)      MSGTEXT
C          EVAL      %SUBST(MSGTEXT: 41: PATHLEN + 4) =
C                                %SUBST(PATH: 1: PATHLEN + 4)
C          EXSR      SNDMSG
C          ENDIF
C*
C          SETON                                           LR
C*
C*****
C* Subroutine to send a message
C*****
C          SNDMSG     BEGSR
C          CALL      'QMHSNDPM'
C          PARM      MESSAGEID
C          PARM      MESSAGEFILE
C          PARM      MSGTEXT
C          PARM      MSGLENGTH
C          PARM      MSGTYPE
C          PARM      STACKENTRY
C          PARM      STACKCOUNTER
C          PARM      MSGKEY
C          PARM      ERRCODE
C          ENDSR
C*
**
| The input file could not be opened.
| There was an error reading from the file.
| The length of the certificate is not valid.
| The certificate is not valid.
| CSNBOWH failed with return/reason codes 9999/9999.
| The output file could not be opened.
| The certified token was written to file

```



```

extern void QDCXLATE(decimal(5,0), char *, char*, char *);
#pragma linkage (QDCXLATE, OS, nowiden)

int main(int argc, char *argv[])
{
/*-----*/
/* Declares for CCA parameters */
/*-----*/
long return_code = 0;
long reason_code = 0;
long exit_data_length = 0;
char exit_data[4];
char rule_array[24];
long rule_array_count;
long token_len = 2500;
char token[2500];
long cloneInfoKeyLength = 500;
unsigned char cloneInfoKey[500];
long cloneInfoLength = 400;
unsigned char cloneInfo[400];
long shareIdx;
char name[64];
char SAname[64];
/*-----*/
/* Declares for working with a PKA token */
/*-----*/
long pub_sec_len; /* Public section length */
long prv_sec_len; /* Private section length */
long cert_sec_len; /* Certificate section length */
long info_subsec_len; /* Information subsection length */
long offset; /* Offset into token */
long tempOffset; /* (Another) Offset into token */
long tempLength; /* Length variable */
long tempLen1, tempLen2; /* temporary length variables */

char cloneShare[] = "cloneShare00"; /* Base cloning share filename */
long count; /* Number of bytes read in from file */
decimal(15,5) shareParm; /* Packed 15 5 var used for converting
/* from packed 15 5 to binary. Numeric
/* parms on system are passed as dec 15 5*/
FILE *fp; /* File pointer

if (argc < 5) /* Check the number of parameters passed */
{
printf("Need to Share index, Sender name, SA name, and cert\n");
return 1;
}

/* Convert the packed decimal 15 5 parm
/* to binary.
memcpy(&shareParm,argv[1],sizeof(shareParm));
shareIdx = shareParm;
memset(name,' ',64); /* Copy the Private key name parm to a
memcpy(name,argv[2],strlen(argv[2])); /* 64 byte space padded var.
memset(SAname,' ',64); /* Copy the Share Admin name parm to a
memcpy(SAname,argv[3],strlen(argv[3]));/* 64 byte space padded var.

fp = fopen(argv[4],"rb"); /* Open the file containing the token
if (!fp)
{
printf("File %s not found.\n",argv[4]);
return 1;
}

memset(token,0,2500); /* Read the token from the file
count = fread(token,1,2500,fp);

```

```

fclose(fp);          /* Close the file */

/* Determine length of token from length */
/* bytes at offset 2 and 3. */
token_len = ((256 * token[2]) + token[3]);
if (count < token_len) /* Check if whole token was read in */
{
    printf("Incomplete token in file\n");
    return 1;
}

/*****/
/* Find the certificate offset in the token */
/* */
/* The layout of the token is */
/* */
/* - Token header - 8 bytes - including 2 length bytes */
/* - Public key section - length bytes at offset 10 overall */
/* - Private key name - 68 bytes */
/* - Certificate section */
/* */
/*****/
pub_sec_len = ((256 * token[10]) + token[11]);

offset = pub_sec_len + 68 + 8; /* Set offset to certificate section */

/* Determine certificate section */
/* length from the length bytes at */
/* offset 2 of the section. */
cert_sec_len = ((256 * token[offset + 2]) + token[offset + 3]);

/*****/
/* Obtain a share */
/*****/
memcpy((void*)rule_array, "OBTAIN ", 8); /* Set rule array */
rule_array_count = 1;

CSUAMKD( &return_code, &reason_code, &exit_data_length,
        exit_data,
        &rule_array_count,
        (unsigned char*)rule_array,
        &shareIdx,
        name,
        SName,
        &cert_sec_len,
        &token[offset],
        &cloneInfoKeyLength,
        cloneInfoKey,
        &cloneInfoLength,
        cloneInfo);

if (return_code != 0)
{
    printf("Master Key Distribution Failed : return reason %d/%d\n",
        return_code, reason_code);
    return 1;
}
else
{
    /*****/
    /* Write signed token out to a file */
    /*****/
    printf("Master Key Distribution worked\n");

/* Build file path name */

```

```

|   if (shareIdx < 9) cloneShare[11] = '0' + shareIdx;
|   else
|   {
|       cloneShare[10] = '1';
|       cloneShare[11] = '0' + shareIdx - 10;
|   }
|
|   fp = fopen(cloneShare,"wb"); /* Open the file */
|   if (!fp)
|   {
|       printf("File %s not be opened for output.¥n",cloneShare);
|       return 1;
|   }
|
|       /* Write out the length of KEK */
|   fwrite((char*)&cloneInfoKeyLength,1,4,fp);
|       /* Write out the KEK */
|   fwrite((char*)cloneInfoKey,1,cloneInfoKeyLength,fp);
|       /* Write out the length of info */
|   fwrite((char*)&cloneInfoLength,1,4,fp);
|       /* Write out the clone info */
|   fwrite((char*)cloneInfo,1,cloneInfoLength,fp);
|   printf("CLone share %d written to %s.¥n",shareIdx,cloneShare);
|
|   fclose(fp); /* Close the file */
|   return 0;
| }
| }

```

例: マスター鍵の共用パーツを取得するための ILE RPG プログラム:

マスター鍵の共用パーツを取得するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* GETSHARE
| D*
| D* Sample program to obtain a master key share as part of the
| D* master key cloning process.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D*       IBM CCA Basic Services Reference and Guide
| D*       (SC31-8609) publication.
| D*
| D* Parameters: Share number
| D*              Name of share sender private key
| D*              Name of certifying key
| D*              Path name of stream file containing receiver certificate
| D*

```

```

| D* Example:
| D* CALL PGM(GETSHARE) PARM(2 SENDR SAKEY RECVR.PUB)
| D*
| D* Use these commands to compile this program on the system:
| D* CRTRPGMOD MODULE(GETSHARE) SRCFILE(SAMPLE)
| D* CRTPGM PGM(GETSHARE) MODULE(GETSHARE)
| D* BNDDIR(QCCA/QC6BNDDIR)
| D*
| D* Note: Authority to the CSUAMKD service program
| D* in the QCCA library is assumed.
| D*
| D* The Common Cryptographic Architecture (CCA) verbs used is
| D* Master_Key_Distribution (CSUAMKD).
| D*
| D*****
| D*-----
| D* Declare variables used by CCA SAPI calls
| D*-----
| D* ** Return code
| DRETURNCODE S 9B 0
| D* ** Reason code
| DREASONCODE S 9B 0
| D* ** Exit data length
| DEXITDATALEN S 9B 0
| D* ** Exit data
| DEXITDATA S 4
| D* ** Rule array count
| DRULEARRAYCNT S 9B 0
| D* ** Rule array
| DRULEARRAY S 16
| D* ** Token length
| DTOKENLEN S 9B 0 INZ(2500)
| D* ** Token and array for subscripting
| DTOKEN DS 2500
| DTOKENARRAY 1 DIM(2500)
| D* ** Private key name
| DPRVNAME S 64
| D* ** Certifying key name
| DCERTKEY S 64
| D*
| DLSTRUCT DS
| D* ** Clone KEK length - one is binary form and the
| D* ** other is used for reading the value from a file
| DCLONEKEKL 9B 0 INZ(500)
| DCLONEKEKLC 1 4
| D* ** Clone info length - one is binary form and the
| D* ** other is used for reading the value from a file
| DCLONEINFOLEN 9B 0 INZ(400)
| DCLONEINFOLENC 5 8
| D* ** Cloning key-encrypting-key
| DCLONEKEK S 500
| D* ** Cloning info
| DCLONEINFO S 400
| D* ** Share index
| DSHAREIDX S 9B 0
| D* ** Data structure for aligning 2 bytes into
| D* ** a 2 bytes integer
| DLENSTRUCT DS 2
| DMSB 1 1
| DLSB 2 2
| DLENGTH 1 2B 0
| D* ** Certificate section length
| DCRTSECLEN S 9B 0
| D* ** Public key section length
| DPUBSECLEN S 9B 0
| D* ** Index into Token array
| DTKNINDEX S 9B 0

```

```

| D*          ** Number of bytes to write out to a file
| DOUTLEN     S          9B 0
| D*          ** File descriptor
| DFILED     S          9B 0
| D*          ** File path and length
| DPSTRUCT   DS
| DPATH      80      INZ(*ALLX'00')
| DSIDX      11      12B 0
| DPATHLEN   S          9B 0
| D*          ** Open Flag - Open for Read only
| DOFLAGR    S          10I 0 INZ(1)
| D*          ** Open flag - Create on open, open for writing,
| D*          ** and clear if exists
| DOFLAGW    S          10I 0 INZ(X'4A')
| D*          ** Base name of file to store cloning share
| DSHAREFILE S          12      INZ('cloneShare00')
| D*
| D*****
| D* Prototype for Master_Key_Distribution (CSUAMKD)
| D*****
| DCSUAMKD    PR
| DRETCOD    9B 0
| DRSNCOD    9B 0
| DEXTDTALN  9B 0
| DEXTDT      4
| DRARRYCT   9B 0
| DRARRY     16
| DSHRINDEX  9B 0
| DKYNAM     64
| DCRTKYNAM  64
| DCRTL      9B 0
| DCRT       2500   OPTIONS(*VARSIZE)
| DCLNKEKL   9B 0
| DCLNKEK    1200   OPTIONS(*VARSIZE)
| DCLNL      9B 0
| DCLN       400    OPTIONS(*VARSIZE)
| D*
| D*****
| D* Prototype for open()
| D*****
| D* value returned = file descriptor (OK), -1 (error)
| Dopen      PR          9B 0 EXTPROC('open')
| D* path name of file to be opened.
| D          128      OPTIONS(*VARSIZE)
| D* Open flags
| D          9B 0 VALUE
| D* (OPTIONAL) mode - access rights
| D          10U 0 VALUE OPTIONS(*NOPASS)
| D* (OPTIONAL) codepage
| D          10U 0 VALUE OPTIONS(*NOPASS)
| D*
| D*****
| D* Prototype for write()
| D*****
| D* value returned = number of bytes written, or -1
| Dwrite     PR          9B 0 EXTPROC('write')
| D* File descriptor returned from open()
| D          9B 0 VALUE
| D* Output buffer
| D          2500   OPTIONS(*VARSIZE)
| D* Length of data to be written
| D          9B 0 VALUE
| D*
| D*****
| D* Prototype for read()
| D*****
| D* value returned = number of bytes actually read, or -1

```

```

Dread          PR          9B 0 EXTPROC('read')
D*   File descriptor returned from open()
D          9B 0 VALUE
D*   Input buffer
D          2500   OPTIONS(*VARSIZE)
D*   Length of data to be read
D          9B 0 VALUE
D*
D*****
D* Prototype for close()
D*****
D*   value returned = 0 (OK), or -1
Dclose        PR          9B 0 EXTPROC('close')
D*   File descriptor returned from open()
D          9B 0 VALUE
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
DMSG          S          75   DIM(6) CTDATA PERRCD(1)
DMSGLENGTH   S          9B 0 INZ(80)
D
DMSGTEXT     1          80
DSAPI        1          7
DFAILRETC    41         44
DFAILRSNC    46         49
DMESSAGEID   S          7   INZ(' ')
DMESSAGEFILE S          21  INZ(' ')
DMSGKEY      S          4   INZ(' ')
DMSGTYPE     S          10  INZ('*INFO ')
DSTACKENTRY  S          10  INZ('* ')
DSTACKCOUNTER S         9B 0 INZ(2)
DERRCODE     DS
DBYTESIN     1          4B 0 INZ(0)
DBYTESOUT    5          8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM *
C* *
C   *ENTRY      PLIST
C               PARM          SINDEXT          15 5
C               PARM          PRVKEY          32
C               PARM          SAKEY           32
C               PARM          FILEPARM        32
C*****
C* Open certificate file
C*****
C* *-----*
C* ** Build path name *
C* *-----*
C          EVAL          PATHLEN = %LEN(%TRIM(FILEPARM))
C   PATHLEN     SUBST     FILEPARM:1   PATH
C* *-----*
C* * Open the file *
C* *-----*
C          EVAL          FILED = open(PATH: OFLAGR)
C* *-----*
C* * Check if open worked *
C* *-----*
C   FILED       IFEQ      -1
C* *-----*
C* * Open failed, send an error message *
C* *-----*
C               MOVEL      MSG(1)          MSGTEXT
C               EXSR       SNDMSG
C               RETURN

```

```

C*
C          ENDIF
C* -----*
C* * Open worked, read certificate and close file *
C* -----*
C          EVAL      TOKENLEN = read(FILED: TOKEN: TOKENLEN)
C          CALLP     close      (FILED)
C*
C* -----*
C* * Check if read operation was OK *
C* -----*
C  TOKENLEN      IFEQ      -1
C                  MOVEL    MSG(2)      MSGTEXT
C                  EXSR     SNDMSG
C                  ENDIF
C*
C* -----*
C* * Check if certificate length is valid *
C* * The length bytes start at position 3 *
C* -----*
C                  EVAL     MSB = TOKENARRAY(3)
C                  EVAL     LSB = TOKENARRAY(4)
C  LENGTH        IFLT     TOKENLEN
C*
C* -----*
C* * Certificate length is not valid *
C* -----*
C                  MOVEL    MSG(3)      MSGTEXT
C                  EXSR     SNDMSG
C                  RETURN
C                  ENDIF
C*
C*****
C* Find the certificate in the token
C*
C* The layout of the token is
C*
C* - Token header - 8 bytes - including 2 length bytes
C* - Public key section - length bytes at position 3 (11 overall)
C* - Private key name - 68 bytes
C* - Certificate section
C*
C* Note: 1 is added because RPG arrays start at 1.
C*****
C          EVAL     MSB = TOKENARRAY(11)
C          EVAL     LSB = TOKENARRAY(12)
C          EVAL     PUBSECLN = LENGTH
C          EVAL     TKNINDEX = PUBSECLN + 68 + 8 + 1
C*
C* -----*
C* * Determine length of certificate section *
C* * Length bytes are at position 2 of the *
C* * section.
C* -----*
C          EVAL     MSB = TOKENARRAY(TKNINDEX + 2)
C          EVAL     LSB = TOKENARRAY(TKNINDEX + 3)
C          EVAL     CRTSECLN = LENGTH
C*
C*****
C* Obtain a certificate
C*****
C* -----*
C* * Set share index number *
C* * (Convert from packed 15 5 to binary) *
C* -----*
C          Z-ADD    SINDEX      SHAREIDX
C*
C* -----*
C* * Set private key name *

```



```

C* *-----*
C          EVAL      LENGTH = %LEN(%TRIM(PRVKEY))
C  LENGTH      SUBST  PRVKEY:1      PRVNAME
C* *-----*
C* * Set certifying key name *
C* *-----*
C          EVAL      LENGTH = %LEN(%TRIM(SAKEY))
C  LENGTH      SUBST  SAKEY:1      CERTKEY
C* *-----*
C* * Set the keywords in the rule array *
C* *-----*
C          MOVE      'OBTAIN '      RULEARRAY
C          Z-ADD     1                RULEARRAYCNT
C* *-----*
C* * Call Master Key Distribution SAPI *
C* *-----*
C          CALLP     CSUAMKD          (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     SHAREIDX:
C                                     PRVNAME:
C                                     CERTKEY:
C                                     CRTSECLN:
C                                     TOKENARRAY(TKNINDEX):
C                                     CLONEKEKL:
C                                     CLONEKEK:
C                                     CLONEINFOLEN:
C                                     CLONEINFO)
C* *-----*
C* * Check the return code *
C* *-----*
C  RETURNCODE  IFGT      0
C* *-----*
C* * Send failure message *
C* *-----*
C          MOVE      MSG(4)          MSGTEXT
C          MOVE      RETURNCODE      FAILRETC
C          MOVE      REASONCODE      FAILRSNC
C          MOVE      'CSUAMKD'      SAPI
C          EXSR      SNDMSG
C          RETURN
C          ENDIF
C*
C*****
C* Write share out to a file
C*****
C* ** Build path name
C          MOVE      *ALLX'00'      PATH
C          MOVE      SHAREFILE      PATH
C  SIDX          ADD      SHAREIDX  SIDX
C  SHAREIDX      IFGE      10
C  SIDX          ADD      246        SIDX
C          ENDIF
C*
C* ** Open the file
C*
C          EVAL      FILED = open(PATH: OFLAGW)
C*
C* ** Check if open worked
C*
C  FILED          IFEQ      -1
C*
C* ** Open failed, send an error message
C*

```

```

|      C          MOVEL   MSG(5)   MSGTEXT
|      C          EXSR    SNDMSG
|      C*
|      C          ELSE
|      C*
|      C          ** Open worked, write certificate out to file and close file
|      C*
|      C          Z-ADD   4          OUTLEN
|      C          CALLP  write      (FILED:
|      C                                     CLONEKEKLC:
|      C                                     OUTLEN)
|      C          CALLP  write      (FILED:
|      C                                     CLONEKEK:
|      C                                     CLONEKEKL)
|      C          CALLP  write      (FILED:
|      C                                     CLONEINFOLENC:
|      C                                     OUTLEN)
|      C          CALLP  write      (FILED:
|      C                                     CLONEINFO:
|      C                                     CLONEINFOLEN)
|      C          CALLP  close     (FILED)
|      C*
|      C          ** Send completion message
|      C*
|      C          MOVEL   MSG(6)   MSGTEXT
|      C          EVAL   %SUBST(MSGTEXT: 32: 12) =
|      C                                     %SUBST(PATH: 1: 12)
|      C          EXSR    SNDMSG
|      C          ENDIF
|      C*
|      C          SETON                                     LR
|      C*
|      C*****
|      C* Subroutine to send a message
|      C*****
|      C          SNDMSG   BEGSR
|      C          CALL     'QMHSNDPM'
|      C          PARM     MESSAGEID
|      C          PARM     MESSAGEFILE
|      C          PARM     MSGTEXT
|      C          PARM     MSGLENGTH
|      C          PARM     MSGTYPE
|      C          PARM     STACKENTRY
|      C          PARM     STACKCOUNTER
|      C          PARM     MSGKEY
|      C          PARM     ERRCODE
|      C          ENDSR
|      C*

```

```

|      **
|      The input file could not be opened.
|      There was an error reading from the file.
|      The length of the certificate is not valid.
|      CSUAMKD failed with return/reason codes 9999/9999.
|      The output file could not be opened.
|      The share was written to file

```

例: マスター鍵の共用パーツをインストールするための ILE C プログラム:

マスター鍵の共用パーツをインストールするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

|      /*-----*/
|      /* PUTSHARE */
|      /* */

```

```

| /* Sample program to install a master key share as part of the */
| /* master key cloning process. */
| /* */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999, 1999 */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: Share number */
| /* Name of share receiver private key */
| /* Name of certifying key */
| /* Stream file containing sender certificate */
| /* */
| /* */
| /* Example: */
| /* CALL PGM(PUTSHARE) PARM(2 RECVR SAKEY SNDR.PUB) */
| /* */
| /* */
| /* Note: This program assumes the card with the profile is */
| /* already identified either by defaulting to the CRP01 */
| /* device or by being explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verbs used is */
| /* Master_Key_Distribution (CSUAMKD). */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(PUTSHARE) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(PUTSHARE) MODULE(PUTSHARE) */
| /* BNDDIR(QCCA/QC6BNDDIR) */
| /* */
| /* Note: Authority to the CSUAMKD service program */
| /* in the QCCA library is assumed. */
| /* */
| /*-----*/
| #include <stdio.h>
| #include <string.h>
| #include "csucinl.h"
| #include "decimal.h"
|
| extern void QDCXLATE(decimal(5,0), char *, char*, char *);
| #pragma linkage (QDCXLATE, OS, nowiden)
|
| int main(int argc, char *argv[])
| {
| /*-----*/
| /* Declares for CCA parameters */
| /*-----*/
| long return_code = 0;
| long reason_code = 0;
| long exit_data_length = 0;
| char exit_data[4];

```

```

| char rule_array[24];
| long rule_array_count;
| long token_len = 2500;
| char token[2500];
| long cloneInfoKeyLength = 500;
| unsigned char cloneInfoKey[500];
| long cloneInfoLength = 400;
| unsigned char cloneInfo[400];
| long shareIdx;
| char name[64];
| char SAname[64];
| /*-----*/
| /* Declares for working with a PKA token */
| /*-----*/
| long pub_sec_len; /* Public section length */
| long prv_sec_len; /* Private section length */
| long cert_sec_len; /* Certificate section length */
| long info_subsec_len; /* Information subsection length */
| long offset; /* Offset into token */
| long tempOffset; /* (Another) Offset into token */
| long tempLength; /* Length variable */
| long tempLen1, tempLen2; /* temporary length variables */
|
| char cloneShare[] = "cloneShare00"; /* Base cloning share filename */
| long count; /* Number of bytes read in from file */
| decimal(15,5) shareParm; /* Packed 15 5 var used for converting */
| /* from packed 15 5 to binary. Numeric */
| /* parms on system are passed as dec 15 5 */
| FILE *fp; /* File pointer */
|
| if (argc < 5) /* Check number of parameters passed in */
| {
| printf("Need Share index, Receiver name, SA name, and cert¥n");
| return 1;
| }
|
| /* Convert the packed decimal 15 5 parm */
| /* to binary. */
| memcpy(&shareParm,argv[1],sizeof(shareParm));
| shareIdx = shareParm;
| memset(name,' ',64); /* Copy the Private key name parm to a */
| memcpy(name,argv[2],strlen(argv[2])); /* 64 byte space padded var. */
| memset(SAname,' ',64); /* Copy the Share Admin name parm to a */
| memcpy(SAname,argv[3],strlen(argv[3])); /* 64 byte space padded var. */
|
| fp = fopen(argv[4],"rb"); /* Open the file containing the token */
| if (!fp)
| {
| printf("File %s not found.¥n",argv[4]);
| return 1;
| }
|
| memset(token,0,2500); /* Read the token from the file */
| count = fread(token,1,2500,fp);
|
| fclose(fp); /* Close the file */
|
| /* Determine length of token from length */
| /* bytes at offset 2 and 3. */
| token_len = ((256 * token[2]) + token[3]);
| if (count < token_len) /* Check if whole token was read in */
| {
| printf("Incomplete token in file¥n");
| return 1;
| }
|
| /*****/

```

```

| /* Find the certificate offset in the token          */
| /*                                                  */
| /* The layout of the token is                    */
| /*                                                  */
| /* - Token header - 8 bytes - including 2 length bytes */
| /* - Public key section - length bytes at offset 10 overall */
| /* - Private key name - 68 bytes                  */
| /* - Certificate section                          */
| /*                                                  */
| /*******/
| pub_sec_len = ((256 * token[10]) + token[11]);
|
| offset = pub_sec_len + 68 + 8; /* Set offset to certificate section */
|
| /* Determine certificate section          */
| /* length from the length bytes at      */
| /* offset 2 of the section.             */
| cert_sec_len = ((256 * token[offset + 2]) + token[offset + 3]);
|
| /*******/
| /* Open and read the clone file          */
| /*******/
| /* Build path name from the base        */
| /* file name and the index              */
| if (shareIdx < 9) cloneShare[11] = '0' + shareIdx;
| else
| {
|     cloneShare[10] = '1';
|     cloneShare[11] = '0' + shareIdx - 10;
| }
|
| fp = fopen(cloneShare,"rb"); /* Open the file with the share */
| if (!fp)
| {
|     printf("Clone share file %s not found.\n",cloneShare);
|     return 1;
| }
|
| /* Read in the length of the KEK          */
| count = fread((char*)&cloneInfoKeyLength,1,4,fp);
|
| if (count < 4) /* Check if there was an error */
| {
|     printf("Clone share file %s contains invalid data.\n",
|           cloneShare);
|     fclose(fp);
|     return 1;
| }
|
| /* Read in the Key encrypting key        */
| count = fread((char*)cloneInfoKey,1,cloneInfoKeyLength,fp);
|
| if (count < cloneInfoKeyLength) /* Check for an error reading */
| {
|     printf("Clone share file %s contains invalid data.\n",
|           cloneShare);
|     fclose(fp);
|     return 1;
| }
|
| /* Read in the length of the clone info  */
| count = fread((char*)&cloneInfoLength,1,4,fp);
|
| if (count < 4) /* Check for an error */
| {
|     printf("Clone share file %s contains invalid data.\n",
|           cloneShare);
|     fclose(fp);
| }

```

```

|     return 1;
| }
|
|         /* Read in the clone info                               */
| count = fread((char*)cloneInfo,1,cloneInfoLength,fp);
|
| if (count < cloneInfoLength) /* Check for an error           */
| {
|     printf("Clone share file %s contains invalid data.¥n",
|           cloneShare);
|     fclose(fp);
|     return 1;
| }
|
| fclose(fp);          /* Close the file                         */
|
| /*****
| /* Install the share                                          */
| /*****
| memcpy((void*)rule_array,"INSTALL ",8); /* Set rule array   */
| rule_array_count = 1;
|
| CSUAMKD( &return_code, &reason_code, &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char*)rule_array,
|         &shareIdx,
|         name,
|         SAname,
|         &cert_sec_len,
|         &token[offset],
|         &cloneInfoKeyLength,
|         cloneInfoKey,
|         &cloneInfoLength,
|         cloneInfo);
|
| if (return_code > 4 )
| {
|     printf("Master Key Distribution Failed : return reason %d/%d¥n",
|           return_code, reason_code);
|     return 1;
| }
| else
| {
|     printf("Master Key share %d successfully installed.¥n",shareIdx);
|     printf("Return reason codes %d/%d¥n",return_code, reason_code);
|     return 0;
| }
| }

```

例: マスター鍵の共用パーツをインストールするための ILE RPG プログラム:

マスター鍵の共用パーツをインストールするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* PUTSHARE
| D*
| D* Sample program to install a master key share as part of
| D* the master key cloning process.
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000

```

```

D*
D* This material contains programming source code for your
D* consideration. These example has not been thoroughly
D* tested under all conditions. IBM, therefore, cannot
D* guarantee or imply reliability, serviceability, or function
D* of these programs. All programs contained herein are
D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
D* these programs and files.
D*
D*
D* Note: Input format is more fully described in Chapter 2 of
D* IBM CCA Basic Services Reference and Guide
D* (SC31-8609) publication.
D*
D* Parameters: Share number
D* Name of share receiver private key
D* Name of certifying key
D* Path name of stream file containing sender certificate
D*
D* Example:
D* CALL PGM(PUTSHARE) PARM(2 RECVR SAKEY SENDER.PUB)
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(PUTSHARE) SRCFILE(SAMPLE)
D* CRTPGM PGM(PUTSHARE) MODULE(PUTSHARE)
D* BNDDIR(QCCA/QC6BNDDIR)
D*
D* Note: Authority to the CSUAMKD service program
D* in the QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used is
D* Master_Key_Distribution (CSUAMKD).
D*
D*****
D*-----
D* Declare variables used by CCA SAPI calls
D*-----
D* ** Return code
DRETURNCODE S 9B 0
D* ** Reason code
DREASONCODE S 9B 0
D* ** Exit data length
DEXITDATALEN S 9B 0
D* ** Exit data
DEXITDATA S 4
D* ** Rule array count
DRULEARRAYCNT S 9B 0
D* ** Rule array
DRULEARRAY S 16
D* ** Token length
DTOKENLEN S 9B 0 INZ(2500)
D* ** Token and array for subscribing
DTOKEN DS 2500
DTOKENARRAY S 1 DIM(2500)
D* ** Private key name
DPRVNAME S 64
D* ** Certifying key name
DCERTKEY S 64
D*
DLSTRUCT DS
D* ** Clone KEK length - one is binary form and the
D* ** other is used for reading the value from a file
DCLONEKEKL S 9B 0 INZ(500)
DCLONEKEKLC S 1 4
D* ** Clone info length - one is binary form and the

```

```

|      D*          ** other is used for reading the value from a file
|      DCLONEINFOLEN          9B 0 INZ(400)
|      DCLONEINFOLENC          5      8
|      D*          ** Cloning key-encrypting-key
|      DCLONEKEK          S          500
|      D*          ** Cloning info
|      DCLONEINFO          S          400
|      D*          ** Share index
|      DSHAREIDX          S          9B 0
|      D*          ** Data structure for aligning 2 bytes into
|      D*          ** a 2 bytes integer
|      DLENSTRUCT          DS          2
|      DMSB          1      1
|      DLSB          2      2
|      DLENGTH          1      2B 0
|      D*          ** Certificate section length
|      DCRTSECLEN          S          9B 0
|      D*          ** Public key section length
|      DPUBSECLEN          S          9B 0
|      D*          ** Index into Token array
|      DTKNINDEX          S          9B 0
|      D*          ** Number of bytes to read from a file
|      DINLEN          S          9B 0
|      D*          ** File descriptor
|      DFILED          S          9B 0
|      D*          ** File path and length
|      DPSTRUCT          DS
|      DPATH          80      INZ(*ALLX'00')
|      DSIDX          11      12B 0
|      DPATHLEN          S          9B 0
|      D*          ** Open Flag - Open for Read only
|      DOFLAGR          S          10I 0 INZ(1)
|      D*          ** Base name of file to store cloning share
|      DSHAREFILE          S          12      INZ('cloneShare00')
|      D*
|      D*****
|      D* Prototype for Master_Key_Distribution (CSUAMKD)
|      D*****
|      DCSUAMKD          PR
|      DRETCOD          9B 0
|      DRSNCOD          9B 0
|      DEXTDTALN          9B 0
|      DEXTDT          4
|      DRARRYCT          9B 0
|      DRARRY          16
|      DSHRINDX          9B 0
|      DKYNAM          64
|      DCRTKYNAM          64
|      DCRTL          9B 0
|      DCRT          2500      OPTIONS(*VARSIZE)
|      DCLNKEKL          9B 0
|      DCLNKEK          1200      OPTIONS(*VARSIZE)
|      DCLNL          9B 0
|      DCLN          400      OPTIONS(*VARSIZE)
|      D*
|      D*****
|      D* Prototype for open()
|      D*****
|      D* value returned = file descriptor (OK), -1 (error)
|      Dopen          PR          9B 0 EXTPROC('open')
|      D* path name of file to be opened.
|      D          128      OPTIONS(*VARSIZE)
|      D* Open flags
|      D          9B 0 VALUE
|      D* (OPTIONAL) mode - access rights
|      D          10U 0 VALUE OPTIONS(*NOPASS)
|      D* (OPTIONAL) codepage

```



```

D          10U 0 VALUE OPTIONS(*NOPASS)
D*
D*****
D* Prototype for read()
D*****
D*   value returned = number of bytes actually read, or -1
Dread          PR          9B 0 EXTPROC('read')
D*   File descriptor returned from open()
D          9B 0 VALUE
D*   Input buffer
D          2500   OPTIONS(*VARSIZE)
D*   Length of data to be read
D          9B 0 VALUE
D*
D*****
D* Prototype for close()
D*****
D*   value returned = 0 (OK), or -1
Dclose         PR          9B 0 EXTPROC('close')
D*   File descriptor returned from open()
D          9B 0 VALUE
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSDPM API
D*-----
DMSG          S          75   DIM(7) CTDATA PERRCD(1)
D          DS
DMSGTEXT          1      80
DSAPI            1      7
DFAILRETC        41     44
DFAILRSNC        46     49
DMSGLENGTH       S          9B 0 INZ(80)
DMESSAGEID       S          7   INZ(' ')
DMESSAGEFILE     S          21  INZ(' ')
DMSGKEY          S          4   INZ(' ')
DMSGTYPE         S          10  INZ('*INFO ')
DSTACKENTRY      S          10  INZ('* ')
DSTACKCOUNTER    S          9B 0 INZ(2)
DERRCODE         DS
DBYTESIN         1      4B 0 INZ(0)
DBYTESOUT        5      8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM *
C* *
C   *ENTRY      PLIST
C              PARM          SINDEXT          15 5
C              PARM          PRVKEY          32
C              PARM          SAKEY           32
C              PARM          FILEPARM        32
C*****
C* Open certificate file
C*****
C* *-----*
C* ** Build path name *
C* *-----*
C          EVAL          PATHLEN = %LEN(%TRIM(FILEPARM))
C   PATHLEN      SUBST    FILEPARM:1   PATH
C* *-----*
C* * Open the file *
C* *-----*
C          EVAL          FILED = open(PATH: OFLAGR)
C* *-----*
C* * Check if open worked *
C* *-----*
C   FILED          IFEQ          -1

```

```

C* *-----*
C* * Open failed, send an error message *
C* *-----*
C          MOVEL      MSG(1)      MSGTEXT
C          EXSR       SNDMSG
C          RETURN
C*
C          ENDIF
C* *-----*
C* * Open worked, read certificate from file and close file *
C* *-----*
C          EVAL      TOKENLEN = read(FILED: TOKEN: TOKENLEN)
C          CALLP     close          (FILED)
C*
C* *-----*
C* * Check if read operation was OK *
C* *-----*
C  TOKENLEN      IFEQ      -1
C          MOVEL      MSG(2)      MSGTEXT
C          EXSR       SNDMSG
C          ENDIF
C*
C* *-----*
C* * Check if certificate length is valid *
C* * The length bytes start at position 3 *
C* *-----*
C          EVAL      MSB = TOKENARRAY(3)
C          EVAL      LSB = TOKENARRAY(4)
C  LENGTH        IFLT      TOKENLEN
C*
C* *-----*
C* * Certificate length is not valid *
C* *-----*
C          MOVEL      MSG(3)      MSGTEXT
C          EXSR       SNDMSG
C          RETURN
C          ENDIF
C*
C*****
C* Find the certificate in the token
C*
C* The layout of the token is
C*
C* - Token header - 8 bytes - including 2 length bytes
C* - Public key section - length bytes at position 2 (11 overall)
C* - Private key name - 68 bytes
C* - Certificate section
C*
C* Note: 1 is added because RPG arrays start at 1.
C*****
C          EVAL      MSB = TOKENARRAY(11)
C          EVAL      LSB = TOKENARRAY(12)
C          EVAL      PUBSECLN = LENGTH
C          EVAL      TKNINDEX = PUBSECLN + 68 + 8 + 1
C*
C* *-----*
C* * Determine length of certificate section *
C* * Length bytes are at position 2 of the *
C* * section.
C* *-----*
C          EVAL      MSB = TOKENARRAY(TKNINDEX + 2)
C          EVAL      LSB = TOKENARRAY(TKNINDEX + 3)
C          EVAL      CRTSECLN = LENGTH
C*
C*****
C* Open and read the clone file
C*****
C* *-----*

```

```

C* * Set share index number *
C* * (Convert from packed 15 5 to binary) *
C* *-----*
C          Z-ADD      SINDEIX      SHAREIDX
C* ** Build path name
C          MOVE      *ALLX'00'      PATH
C          MOVE      SHAREFILE      PATH
C* ** Adjust two digits on file name by adding to their
C* ** character value
C          SIDX      ADD      SHAREIDX      SIDX
C* ** If the index is greater than or equal to 10
C* ** then add 246 to force the first character to change
C          SHAREIDX  IFGE      10
C          SIDX      ADD      246          SIDX
C          ENDIF
C*
C* ** Open the file
C*
C          EVAL      FILED = open(PATH: OFLAGR)
C*
C* ** Check if open worked
C*
C          FILED      IFEQ      -1
C*
C* ** Open failed, send an error message
C*
C          MOVE      MSG(4)          MSGTEXT
C          EXSR      SNDMSG
C*
C          ELSE
C*
C* ** Open worked, read in the clone information and close file
C*
C          SETON                                01
C          Z-ADD      4          INLEN
C          EVAL      INLEN = read(FILED: CLONEKEKLC: INLEN)
C*
C* *-----*
C* * Check if read operation was OK *
C* *-----*
C          INLEN      IFNE      4
C          MOVE      MSG(5)          MSGTEXT
C          EXSR      SNDMSG
C          SETOFF                                01
C          ENDIF
C*
C          01          EVAL      INLEN = read(FILED: CLONEKEK: CLONEKEKL)
C*
C          01INLEN      IFNE      CLONEKEKL
C          MOVE      MSG(5)          MSGTEXT
C          EXSR      SNDMSG
C          SETOFF                                01
C          ENDIF
C*
C          01          Z-ADD      4          INLEN
C          01          EVAL      INLEN = read(FILED: CLONEINFOLENC: INLEN)
C*
C* *-----*
C* * Check if read operation was OK *
C* *-----*
C          01INLEN      IFNE      4
C          MOVE      MSG(5)          MSGTEXT
C          EXSR      SNDMSG
C          SETOFF                                01
C          ENDIF
C*
C          01          EVAL      INLEN = read(FILED: CLONEINFO: CLONEINFOLEN)

```

```

C*
C* *-----*
C* * Check if read operation was OK *
C* *-----*
C 01INLEN      IFNE      CLONEINFOLEN
C              MOVE      MSG(5)      MSGTEXT
C              EXSR      SNDMSG
C              SETOFF
C              ENDIF
C*
C              CALLP      close      (FILED)
C N01          SETON
C*
C*****
C* Obtain a certificate
C*****
C* *-----*
C* * Set share index number *
C* *-----*
C              Z-ADD      SINDEK      SHAREIDX
C* *-----*
C* * Set private key name *
C* *-----*
C              EVAL      LENGTH = %LEN(%TRIM(PRVKEY))
C              LENGTH    SUBST      PRVKEY:1      PRVNAME
C* *-----*
C* * Set certifying key name *
C* *-----*
C              EVAL      LENGTH = %LEN(%TRIM(SAKEY))
C              LENGTH    SUBST      SAKEY:1      CERTKEY
C* *-----*
C* * Set the keywords in the rule array *
C* *-----*
C              MOVE      'INSTALL '      RULEARRAY
C              Z-ADD      1              RULEARRAYCNT
C* *-----*
C* * Call Master Key Distribution SAPI *
C* *-----*
C              CALLP      CSUAMKD      (RETURNCODE:
C                                  REASONCODE:
C                                  EXITDATALEN:
C                                  EXITDATA:
C                                  RULEARRAYCNT:
C                                  RULEARRAY:
C                                  SHAREIDX:
C                                  PRVNAME:
C                                  CERTKEY:
C                                  CRTSECLN:
C                                  TOKENARRAY(TKNINDEX):
C                                  CLONEKEKL:
C                                  CLONEKEK:
C                                  CLONEINFOLEN:
C                                  CLONEINFO)
C* *-----*
C* * Check the return code *
C* *-----*
C              RETURNCODE  IFGT      4
C* *-----*
C* * Send failure message *
C* *-----*
C              MOVE      MSG(6)      MSGTEXT
C              MOVE      RETURNCODE  FAILRET
C              MOVE      REASONCODE  FAILRSNC
C              MOVE      'CSUAMKD'   SAPI
C              EXSR      SNDMSG
C              RETURN
C              ENDIF

```

```

| C* *-----*
| C* * Send success message *
| C* *-----*
| C          MOVEL      MSG(7)          MSGTEXT
| C          EVAL      %SUBST(MSGTEXT: 32: 12) =
| C                                %SUBST(PATH: 1: 12)
| C          EXSR      SNDMSG
| C          ENDIF
| C*
| C          SETON
| C
| C*
| C*****
| C* Subroutine to send a message
| C*****
| C      SNDMSG      BEGSR
| C          CALL      'QMHSNDPM'
| C          PARM          MESSAGEID
| C          PARM          MESSAGEFILE
| C          PARM          MSGTEXT
| C          PARM          MSGLENGTH
| C          PARM          MSGTYPE
| C          PARM          STACKENTRY
| C          PARM          STACKCOUNTER
| C          PARM          MSGKEY
| C          PARM          ERRCODE
| C          ENDSR
| C*

```

LR

**

```

| The certificate file could not be opened.
| There was an error reading from the certificate file.
| The length of the certificate is not valid.
| The clone share file could not be opened.
| The clone share file either could not be read or has invalid data.
| CSUAMKD failed with return/reason codes 9999/9999.
| The share was successfully installed.

```

例: 保管鍵をリストするための ILE C プログラム:

保管鍵をリストするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| /*-----*/
| /* List the names of the RSA private keys retained. */
| /* */
| /* */
| /* */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000 */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */

```

```

| /* none. */
| /* */
| /* Example: */
| /* CALL PGM(LISTRETAIN) */
| /* */
| /* */
| /* Note: This program assumes the card with the profile is */
| /* already identified either by defaulting to the CRP01 */
| /* device or by being explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Access_Control_Initialization (CSUAACI). */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(LISTRETAIN) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(LISTRETAIN) MODULE(LISTRETAIN) */
| /* BNDSRVPGM(QCCA/CSNDRKL) */
| /* */
| /* Note: Authority to the CSNDRKL service program in the */
| /* QCCA library is assumed. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Retained_Key_List (CSNDRKL). */
| /* */
| /*-----*/
| #include <string.h>
| #include <stdio.h>
| #include "csucincl.h"
|
| void main(void)
| {
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
| long return_code;
| long reason_code;
| long exit_data_length;
| unsigned char exit_data[2];
| long rule_array_count;
| unsigned char rule_array[2][8];
| /*-----*/
| /* CCA parameters unique to CSNDRKL */
| /*-----*/
| unsigned char key_label_mask[64];
| unsigned char key_label[500][64];
| long retain_key_count;
| long key_label_count = 500;
| int k;
|
| /*-----*/
| /* Set up label mask, ie. which key name to retrieve. */
| /* *.*.*.*.*.* is a wildcard for all keys. */
| /*-----*/
| memset(key_label, 0x00, sizeof(key_label) );
| memset(key_label_mask, ' ', sizeof(key_label_mask));
| memcpy(key_label_mask,"*.*.*.*.*.*",13);
| rule_array_count = 0;
|
| /*-----*/
| /* Invoke the verb to get the list of the retained keys. */
| /*-----*/
| CSNDRKL(&return_code,
| &reason_code,

```

```

|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char*)rule_array,
|         key_label_mask,
|         &retain_key_count,
|         &key_label_count,
|         (unsigned char*)key_label);
|
| /*-----*/
| /* Check the results */
| /*-----*/
| if (return_code != 0)
| {
|     printf("Retained Key List failed with return/reason %d/%d %n",
|           return_code, reason_code);
|     return;
| }
| else
| {
|     /*-----*/
|     /* Display number of keys retained/returned. */
|     /*-----*/
|     printf("Retained key count [%d] %n", retain_key_count);
|     printf("No. of key labels returned [%d] %n", key_label_count);
|     if (key_label_count > 0)
|     {
|         /*-----*/
|         /* Display the names of each key returned. */
|         /*-----*/
|         printf("Retain list = %n" );
|         for (k = 0 ;k < key_label_count; k++)
|         {
|             printf( "[%.64s] %n",key_label[k]);
|         }
|     }
| }
| }

```

例: 保管鍵をリストするための ILE RPG プログラム:

保管鍵をリストするには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D*
| D* List the names of the RSA private keys retained within the
| D* .
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of

```

```

|      D*      IBM CCA Basic Services Reference and Guide
|      D*      (SC31-8609) publication.
|      D*
|      D* Parameters: None
|      D*
|      D* Example:
|      D* CALL PGM(LISTRETAIN)
|      D*
|      D* Use these commands to compile this program on the system:
|      D* CRTRPGMOD MODULE(LISTRETAIN) SRCFILE(SAMPLE)
|      D* CRTPGM PGM(LISTRETAIN) MODULE(LISTRETAIN)
|      D*      BNDSRVPGM(QCCA/CSNDRKL)
|      D*
|      D* The Common Cryptographic Architecture (CCA) verbs used are
|      D* Retained_key_List (CSNDRKL)
|      D*
|      D* Note: Authority to the CSNDRKL service program in the
|      D*      QCCA library is assumed.
|      D*
|      D*
|      D* Note: This program assumes the card with the profile is
|      D*      already identified either by defaulting to the CRP01
|      D*      device or by being explicitly named using the
|      D*      Cryptographic_Resource_Allocate verb. Also this
|      D*      device must be varied on and you must be authorized
|      D*      to use this device description.
|      D*
|      D*****
|      D*-----
|      D* Declare variables for CCA SAPI calls
|      D*-----
|      D*      ** Return code
|      DRETURNCODE      S          9B 0
|      D*      ** Reason code
|      DREASONCODE      S          9B 0
|      D*      ** Exit data length
|      DEXITDATALEN      S          9B 0
|      D*      ** Exit data
|      DEXITDATA        S           4
|      D*      ** Rule array count
|      DRULEARRAYCNT     S          9B 0
|      D*      ** Rule array
|      DRULEARRAY        S          16
|      D*      ** Key label mask
|      DKEYLBLMASK       S          64
|      D*      ** Key count
|      DKEYCOUNT        S          9B 0
|      D*      ** Label count
|      DLABELCOUNT     S          9B 0
|      D*      ** Label list and label array
|      DLABELLIST        DS          3200
|      DLABELS           64      DIM(50)
|      D*      ** Loop counter
|      DI                S          9B 0
|      D*
|      D*****
|      D* Prototype for Retained_Key_List
|      D*****
|      DCSNDRKL          PR
|      DRETCODE          9B 0
|      DRSNCODE          9B 0
|      DEXTDTALEN        9B 0
|      DEXTDTA           4
|      DRARRAYCT         9B 0
|      DRARRAY           16
|      DKYLBLMSK         64
|      DKYCOUNT         9B 0

```



```

DLBLCOUNT          9B 0
DLBLS                64
D*
D*-----
D*          ** Declares for sending messages to the
D*          ** job log using the QMHSNDPM API
D*-----
MSG              S          75    DIM(4) CTDATA PERRCD(1)
MSGLENGTH       S          9B 0  INZ(75)
D               DS
MSGTEXT         1          75
DNUMKEYS        1          3
DNUMLABELS      25         26
DDSPLBL         2          65
DFAILRETC       41         44
DFAILRNC        46         49
DMESSAGEID      S          7    INZ('      ')
DMESSAGEFILE    S          21    INZ('      ')
MSGKEY          S          4     INZ('      ')
MSGTYPE         S          10    INZ('*INFO ')
DSTACKENTRY     S          10    INZ('*      ')
DSTACKCOUNTER   S          9B 0  INZ(2)
DERRCODE        DS
DBYTESIN         1          4B 0  INZ(0)
DBYTESOUT        5          8B 0  INZ(0)
D*
C*****
C* START OF PROGRAM *
C* *
C*-----*
C* No rule array keywords *
C*-----*
C          Z-ADD      0          RULEARRAYCNT
C*-----*
C* Get up to 50 labels *
C*-----*
C          Z-ADD      50          LABELCOUNT
C*-----*
C* Set the mask to everything *
C*-----*
C          MOVE      '*'          KEYLBLMASK
C*-----*
C* Call Retained Key List SAPI *
C*-----*
C          CALLP     CSNDRKL      (RETURNCODE:
C                                REASONCODE:
C                                EXITDATALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                KEYLBLMASK:
C                                KEYCOUNT:
C                                LABELCOUNT:
C                                LABELLIST)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      4
C*          *-----*
C*          * Send error message *
C*          *-----*
C          MOVE      MSG(1)      MSGTEXT
C          MOVE      RETURNCODE  FAILRETC
C          MOVE      REASONCODE  FAILRNC
C          EXSR      SNDMSG
C*
C          ELSE

```

```

C*
C* *-----*
C* * Check number of keys *
C* *-----*
C LABELCOUNT IFEQ 0
C* *-----*
C* * Send message saying there are no keys *
C* *-----*
C MOVE MSG(2) MSGTEXT
C EXSR SNDMSG
C
C ELSE
C*
C* *-----*
C* * Send message with number of keys *
C* *-----*
C MOVE MSG(3) MSGTEXT
C MOVE KEYCOUNT NUMKEYS
C MOVE LABELCOUNT NUMLABELS
C EXSR SNDMSG
C*
C* *-----*
C* * Display each key label up to 50 *
C* *-----*
C MOVE MSG(4) MSGTEXT
C FOR I=1 BY 1 TO LABELCOUNT
C MOVE LABELS(I) DSPLBL
C EXSR SNDMSG
C ENDFOR
C*
C ENDF
C ENDF
C*
C SETON LR
C*
C*****
C* Subroutine to send a message
C*****
C SNDMSG BEGSR
C CALL 'QMHSNDPM'
C PARM MESSAGEID
C PARM MESSAGEFILE
C PARM MSGTEXT
C PARM MSGLENGTH
C PARM MSGTYPE
C PARM STACKENTRY
C PARM STACKCOUNTER
C PARM MSGKEY
C PARM ERRCODE
C ENDSR
**
CSNDRKL failed with return/reason codes 9999/9999
There are no retained keys
000 keys were found and 00 labels returned
[ ]

```

例: 保管鍵を削除するための ILE C プログラム:

保管鍵を削除するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

/*-----*/
/* Delete a retained key */
/* */

```

```

| /* */
| /* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000 */
| /* */
| /* This material contains programming source code for your */
| /* consideration. These examples have not been thoroughly */
| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* Note: Input format is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* none. */
| /* */
| /* Example: */
| /* CALL PGM(DLTRNKEY) (SSLPRIV.KEY.ONE) */
| /* */
| /* Note: This program assumes the card with the profile is */
| /* already identified either by defaulting to the CRP01 */
| /* device or by being explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Retained_Key_Delete (CSNDRKD). */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(DLTRNKEY) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(DLTRNKEY) MODULE(DLTRNKEY) */
| /* BNDSRVPGM(QCCA/CSNDRKD) */
| /* */
| /* Note: Authority to the CSNDRKD service program in the */
| /* QCCA library is assumed. */
| /* */
| /*-----*/
| #include <string.h>
| #include <stdio.h>
| #include "csucinl.h"
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define OK 0
| #define WARNING 4
|
| void main(int argc, char * argv[1])
| {
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
| long return_code;
| long reason_code;
| long exit_data_length;
| unsigned char exit_data[2];

```

```

| long          rule_array_count = 0;
| unsigned char rule_array[1][8];
| unsigned char key_label[64];
|
| /*-----*/
| /* Process the parameters */
| /*-----*/
| if (argc < 1)
| {
|     printf("Key label parameter must be specified.\n");
|     return;
| }
|
| /*-----*/
| /* Set up the key label */
| /*-----*/
| memset(key_label, ' ', 64 );
| memcpy(key_label, argv[1], strlen(argv[1]) );
|
| /*-----*/
| /* Call the Retained Key List SAPI */
| /*-----*/
| CSNDRKD(&return_code,
|         &reason_code,
|         &exit_data_length,
|         exit_data,
|         &rule_array_count,
|         (unsigned char*)rule_array,
|         key_label);
|
| /*-----*/
| /* Check the return code and display the results */
| /*-----*/
| if ( (return_code == OK) || (return_code == WARNING) )
| {
|     printf("Request was successful\n");
|     return;
| }
| else
| {
|     printf("Request failed with return/reason codes: %d/%d \n",
|           return_code, reason_code);
|     return;
| }
|
| }

```

例: 保管鍵を削除するための ILE RPG プログラム:

保管鍵を削除するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

```

| D*****
| D* DLTRTNKEY
| D*
| D* Sample program to delete a retained key
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your

```

```

D* consideration. These example has not been thoroughly
D* tested under all conditions. IBM, therefore, cannot
D* guarantee or imply reliability, serviceability, or function
D* of these programs. All programs contained herein are
D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
D* these programs and files.
D*
D*
D* Note: Input format is more fully described in Chapter 2 of
D* IBM CCA Basic Services Reference and Guide
D* (SC31-8609) publication.
D*
D* Parameters:
D* Retained key label name
D* (64 characters - pad with blanks on the right)
D*
D* Example:
D*
D* CALL DLTRTNKEY +
D* 'PKA.RETAINED.KEY.123
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(DLTRTNKEY) SRCFILE(SAMPLE)
D* CRTPGM PGM(DLTRTNKEY) MODULE(DLTRTNKEY)
D* BNDSRVPGM(QCCA/CSNDRKD)
D*
D* Note: Authority to the CSNDRKD service program in the
D* QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Retained_Key_Delete (CSNDRKD)
D*
D*****
D*-----
D* Declare variables for CCA SAPI calls
D*-----
D*          ** Return code
DRETURNCODE S          9B 0
D*          ** Reason code
DREASONCODE S          9B 0
D*          ** Exit data length
DEXITDATALEN S          9B 0
D*          ** Exit data
DEXITDATA S          4
D*          ** Rule array count
DRULEARRAYCNT S          9B 0
D*          ** Rule array
DRULEARRAY S          16
D*          ** Retained key label
DKEYNAME S          64
D*
D*****
D* Prototype for Retained_Key_Delete (CSNDRKD)
D*****
DCSNDRKD PR
DRETCODE          9B 0
DRSNCODE          9B 0
DEXTDTALEN        9B 0
DEXTDTA           4
DRARRAYCT         9B 0
DRARRAY           16
DKEYNAM           64
D*
D*-----
D*          ** Declares for sending messages to the

```

```

D*          ** job log using the QMHSNDPM API
D*-----
DMSG          S          75  DIM(2) CTDATA PERRCD(1)
DMSGLENGTH   S          9B 0 INZ(75)
D            DS
DMSGTEXT      1          75
DFAILMSGTEXT  1          50
DFAILRETC     41         44
DFAILRSNC     46         49
DMESSAGEID    S          7  INZ('      ')
DMESSAGEFILE  S          21 INZ('      ')
DMSGKEY       S          4  INZ('      ')
DMSGTYPE      S          10 INZ('*INFO ')
DSTACKENTRY   S          10 INZ('*      ')
DSTACKCOUNTER S          9B 0 INZ(2)
DERRCODE      DS
DBYTESIN      1          4B 0 INZ(0)
DBYTESOUT     5          8B 0 INZ(0)
D*
C*****
C* START OF PROGRAM *
C* *
C   *ENTRY      PLIST
C               PARM          KEYNAME
C* *
C*-----*
C* Set the keywords in the rule array *
C*-----*
C               Z-ADD      0          RULEARRAYCNT
C*-----*
C* Call Retained Key Delete SAPI *
C*-----*
C               CALLP      CSNDRKD      (RETURNCODE:
C                                     REASONCODE:
C                                     EXITDATALEN:
C                                     EXITDATA:
C                                     RULEARRAYCNT:
C                                     RULEARRAY:
C                                     KEYNAME)
C*-----*
C* Check the return code *
C*-----*
C   RETURNCODE  IFGT      4
C* *-----*
C* * Send error message *
C* *-----*
C               MOVE      MSG(1)      MSGTEXT
C               MOVE      RETURNCODE  FAILRETC
C               MOVE      REASONCODE  FAILRSNC
C               EXSR      SNDMSG
C*
C               ELSE
C* *-----*
C* * Send success message *
C* *-----*
C               MOVE      MSG(2)      MSGTEXT
C               EXSR      SNDMSG
C*
C               ENDIF
C
C               SETON          LR
C*
C*****
C* Subroutine to send a message
C*****
C   SNDMSG      BEGSR
C               CALL          'QMHSNDPM'

```

	C	PARM	MESSAGEID
	C	PARM	MESSAGEFILE
	C	PARM	MSGTEXT
	C	PARM	MSGLENGTH
	C	PARM	MSGTYPE
	C	PARM	STACKENTRY
	C	PARM	STACKCOUNTER
	C	PARM	MSGKEY
	C	PARM	ERRCODE
	C	ENDSR	
	C*		

**

| CSNDRKD failed with return/reason codes 9999/9999'
| The request completed successfully

| 暗号化コプロセッサのトラブルシューティング

| 暗号化コプロセッサで発生する基本的な問題のいくつかに対処するには、以下に示すトラブルシューティング方法を使用します。トラブルシューティング情報がユーザーの問題に対処していない場合には、サービス技術員に連絡してください。

| 関連する製品とプログラムに対して、現行の PTF すべてを適用しているか常時確認してください。

| リターン・コードの使用

| 問題を検出してトラブルシューティングを行うための基本的な方法は、リターン・コードと理由コードを監視することです。

- | • **0** のリターン・コードは、正常に完了したことを示します。追加情報を提供するため、暗号化コプロセッサは、いくつかのゼロ以外の理由コードをこのリターン・コードに関連付けます。
- | • **4** のリターン・コードは、アプリケーション・プログラミング・インターフェース (API) は処理を完了しましたが、異常なイベントが発生したことを示します。アプリケーション・プログラムにより発生した問題に関連している場合、あるいは、API に提供されたデータに基づく通常の発生場合があります。
- | • **8** のリターン・コードは、API が正常に完了しなかったことを示します。おそらくアプリケーション・プログラミング・エラーが原因になっています。
- | • **12** のリターン・コードは、コプロセッサのセットアップまたは構成で発生する問題のいくつかを示しています。このコードは、API の処理が正常に完了しなかったことを示しています。
- | • **16** のリターン・コードは、通常、Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP)、システムのライセンス内部コード、または暗号化コプロセッサのライセンス内部コードの重大エラーを示しています。これらのタイプのエラーについては、サービス技術員に連絡する必要があります。

| ジョブ・ログまたはシステム・オペレーター (QSYSOPR) 待ち行列に表示されるメッセージを分析することで問題のトラブルシューティングを行うこともできます。一般的に、ジョブ・ログにメッセージを送るイベントは、関連したリターン・コードと理由コードを呼び出しプログラミングに戻します。システム・オペレーターに送られたメッセージは、重大問題を報告する場合、通常、問題についての追加情報元を指し示します。このような情報は IBM のサービス技術員を対象としているため、問題判別に必ずしも有効であるとはかぎりません。

| 一般的なエラー

| 次のような一般的なエラーにはよく注意する必要があります。

- 1 | • **装置をオンに変更したか** 装置をオンに変更するまでは、暗号化コプロセッサには、いかなる要求も送

1 | ることができません。
- 1 | • **CCA は装置を検出しているか** Cryptographic_Resource_Allocate API を明示的に使用しなかった場合に

1 | は、暗号装置に CRP01 という名前を付ける必要があります。名前を付けなかった場合には、CCA はど

1 | の装置も選択することができません。装置に CRP01 という名前を付けるか、または

1 | Cryptographic_Resource_Allocate CCA API を使用するようにプログラムを変更して装置を選択します。
- 1 | • **正しい装置を選択しているか** デフォルトの装置 (例えば、CRP01 という名前の装置) と追加装置がある

1 | 場合には、暗号化コプロセッサは、ユーザーが Cryptographic_Resource_Allocate を使用しないかぎり、

1 | デフォルトの装置が選択されます。
- 1 | • **暗号化コプロセッサは鍵ストア・ファイルを検出しているか** Key_Store_Designate SAPI を明示的に使

1 | 用しなかった場合には、CCA CSP サポートは、装置記述で名前が付けられたファイルを使おうとしま

1 | す。装置記述でファイルに名前を付けていない場合には、暗号化コプロセッサはすべてのファイルを

1 | 検出しません。
- 1 | • **マスター鍵をロードして設定しているか** 暗号化コプロセッサは、マスター鍵をロードしない限り、暗

1 | 号化コプロセッサを構成するための要求以外には、いかなる暗号要求も完了しません。
- 1 | • **古いマスター鍵レジスターに鍵が含まれているか** 暗号化コプロセッサは、古いマスター鍵レジスター

1 | に値がないかぎり、現行のマスター鍵では鍵を再暗号化することができません。
- 1 | • **デフォルトの役割には特定のハードウェア・コマンドを使用する権限があるか** 権限がない場合には、正

1 | しい権限のある役割を使用するプロファイルを使用してログオンする必要があります。
- 1 | • **特定のハードウェア・コマンドを使用する権限を持っている役割があるか** 暗号化コプロセッサがハー

1 | ドウェア・コマンドを必要とするが、そのコマンドを使用するための役割を許可していない場合には、

1 | 暗号化コプロセッサを再初期化する必要があります。再初期化は、Cryptographic_Facility_Control API

1 | またはシステム保守ツールにあるハードウェア保守管理機能を使用します。

1 | Cryptographic_Facility_Control API を使用するには、暗号化コプロセッサを再初期化するハードウェ

1 | ア・コマンドに役割を許可する必要があります。このような役割が存在しない場合、ハードウェア保守

1 | 管理機能を使用します。
- 1 | • **機能制御ベクトルがロードされているか** 暗号化コプロセッサは、機能制御ベクトルをロードするまで

1 | は、構成以外の暗号操作を実行することはできません。
- 1 | • **マスター鍵をロードしている場合には、新規のマスター鍵レジスターを削除してから開始したか** 暗号化

1 | コプロセッサが新規のマスター鍵レジスターを部分的にロードしている場合には、マスター鍵の最初

1 | の部分をロードすることはできません。
- 1 | • **実行する権限を DEFAULT の役割から削除する前に、コプロセッサでクロックを設定したか** 設定し

1 | ていない場合、Cryptographic_Facility_Control API またはシステム保守ツールにあるハードウェア・サー

1 | ビス管理を使用して暗号化コプロセッサを初期化します。Cryptographic_Facility_Control API を使用す

1 | るには、暗号化コプロセッサを再初期化するハードウェア・コマンドに役割を許可する必要がありま

1 | す。このような役割が存在しない場合、ハードウェア保守管理機能を使用します。
- 1 | • **公開鍵 - 秘密鍵ペアを生成する前に EID を設定したか** RSA 鍵を生成する前に EID を設定する必要が

1 | あります。
- 1 | • **ヌル鍵トークンの最初のバイトを 2 進 0 に正しく初期化したか** 初期化していない場合、CCA サポー

1 | トは最初のバイトを鍵ラベルとして使用することもあります。CCA サポートは、最初のバイトを不良

1 | ラベル形式としてレポートするか、または鍵レコードを検出できたとレポートします。
- 1 | • **PKA 鍵ストア・ファイルおよび保管されている PKA 鍵にあるラベルに対して同じ名前を使用したか**

1 | 同じ名前を使用した場合には、暗号化コプロセッサは、鍵ストア・ファイルを最初に検索するため、

1 | 保管されている鍵を検出することはありません。

・ スケルトン PKA 鍵トークンのいずれかのフィールドに EBCDIC データがあるか。暗号化コプロセッサは、特に、多くのフィールドで ASCII データかどうかをチェックして、EBCDIC データを検出した場合にはエラーを戻します。

暗号化コプロセッサの再初期化

暗号化コプロセッサを誤ってセットアップした場合には、構成は使用不可になり、すべての暗号機能が実行できなくなります。また、API を使用して回復することもできません。例えば、マスター鍵の設定が許可された役割がなかったり、新規の役割またはプロファイルの変更または作成が許可された役割がない構成となってしまう場合があります。Cryptographic_Facility_Control (CSUACFC) SAPI を使用すると、カードを再初期化するためのハードウェア・コマンドを呼び出すことができます。

しかし、場合によっては、ハードウェア・コマンドに許可されている役割がない場合もあります。その場合には、システム保守ツールのハードウェア保守管理機能で提供されている機能を使用して、ライセンス内部コードを再ロードする必要があります。

暗号化コプロセッサにあるライセンス内部コードの更新

ライセンス内部コードを暗号化コプロセッサにロードすると、暗号化コプロセッサに保管されているマスター鍵、すべての秘密鍵、およびすべての役割とプロファイルが消去されます。したがって、サーバーは暗号化コプロセッサにライセンス内部コードの PTF を自動的にロードしないため、PTF を使用可能にするには、システム管理者側でアクションを取る必要があります。ライセンス内部コードをロードする前に、マスター鍵のハードコピーを作成しておくといった、回復を確実に可能にするための適切なアクションを取ります。

注: マスター鍵をランダムに生成した場合には、その鍵を 2 番目の暗号化コプロセッサに複製する必要があります。複製しない場合、暗号化コプロセッサを再初期化したときに暗号化された鍵がすべて失われます。

関連タスク

282 ページの『ハードウェア・サービス管理プログラムの使用』
ハードウェア保守管理機能は、論理的な観点およびパッケージングの観点から、システムのハードウェアを表示および操作するためのツールであり、入出力 (I/O) プロセッサおよびデバイスのデバッグを支援するものです。また、暗号化コプロセッサを再初期化するためにも使用されます (初期化されていない状態に戻します)。

例: 暗号化コプロセッサを再初期化するための ILE C プログラム

暗号化コプロセッサを再初期化するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

提供されているプログラム例を使用する場合には、必要に応じて変更します。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```
/*-----*/
/* Clear the card (reset to manufactured state).          */
/*                                                         */
/*                                                         */
/* COPYRIGHT 5769-SS1 (C) IBM CORP. 1999                */
/*                                                         */
/* This material contains programming source code for your */
/* consideration. These examples have not been thoroughly */
```

```

| /* tested under all conditions. IBM, therefore, cannot */
| /* guarantee or imply reliability, serviceability, or function */
| /* of these program. All programs contained herein are */
| /* provided to you "AS IS". THE IMPLIED WARRANTIES OF */
| /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE */
| /* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for */
| /* these programs and files. */
| /* */
| /* Note: This verb is more fully described in Chapter 2 of */
| /* IBM CCA Basic Services Reference and Guide */
| /* (SC31-8609) publication. */
| /* */
| /* Parameters: */
| /* none. */
| /* */
| /* Example: */
| /* CALL PGM(REINIT) */
| /* */
| /* Note: This program assumes the device to use is */
| /* already identified either by defaulting to the CRP01 */
| /* device or by being explicitly named using the */
| /* Cryptographic_Resource_Allocate verb. Also this */
| /* device must be varied on and you must be authorized */
| /* to use this device description. */
| /* */
| /* Use these commands to compile this program on the system: */
| /* ADDLIB LIB(QCCA) */
| /* CRTCMOD MODULE(REINIT) SRCFILE(SAMPLE) */
| /* CRTPGM PGM(REINIT) MODULE(REINIT) BNDSRVPGM(QCCA/CSUACFC) */
| /* */
| /* Note: Authority to the CSUACFC service program in the */
| /* QCCA library is assumed. */
| /* */
| /* The Common Cryptographic Architecture (CCA) verb used is */
| /* Cryptographic_Facilities_Control (CSUACFC). */
| /* */
| /*-----*/
|
| #include "csucincl.h" /* header file for CCA Cryptographic */
| /* Service Provider */
| #include <stdio.h>
| #include <string.h>
| #include <stdlib.h>
|
| /*-----*/
| /* standard return codes */
| /*-----*/
|
| #define ERROR -1
| #define OK 0
| #define WARNING 4
|
| #define TOKENSIZE 8 /* number of bytes in random token */
|
| int main(int argc, char *argv[])
| {
| /*-----*/
| /* standard CCA parameters */
| /*-----*/
|
| long return_code = 0;

```

```

|   long reason_code = 0;
|   long exit_data_length = 2;
|   char exit_data[4];
|   char rule_array[2][8];
|   long rule_array_count = 2;
|
|   /*-----*/
|   /* fields unique to this sample program */
|   /*-----*/
|
|   long verb_data_length = TOKENSIZE;
|   char verb_data[TOKENSIZE];
|   char verb_data2[TOKENSIZE];
|   int i;
|
|   /* set keywords in the rule array */
|
|   memcpy(rule_array,"ADAPTER1RQ-TOKEN",16);
|
|   /* get a random token from the card - returned in verb_data */
|
|   CSUACFC( &return_code,
|           &reason_code,
|           &exit_data_length,
|           exit_data,
|           &rule_array_count,
|           (char *)rule_array,
|           &verb_data_length,
|           (char *)verb_data);
|
|   if ( (return_code == OK) | (return_code == WARNING) )
|   {
|   printf("Random token was successfully returned.¥n");
|
|   printf("Return/reason codes ");
|
|   printf("%1d/%1d¥n", return_code, reason_code);
|
|   /* get the one's complement of token and store in verb_data2. */
|   /* operate on one byte at a time */
|
|   for(i = 0; i < TOKENSIZE; i++)
|   {
|       verb_data2[i] = ~verb_data[i];
|   }
|
|   /* change keyword in rule array */
|
|   memcpy(&rule_array[1],"RQ-REINT",8);
|
|   /* invoke the verb to reset the card */
|
|   CSUACFC( &return_code,
|           &reason_code,
|           &exit_data_length,
|           exit_data,
|           &rule_array_count,
|           (char *)rule_array,
|           &verb_data_length,
|           verb_data2);
|
|   if ( (return_code == OK) | (return_code == WARNING) )
|   {
|       printf("card successfully cleared/reset.¥n");
|
|       printf("Return/reason codes ");

```

```

|         printf("%ld/%ld¥n¥n", return_code, reason_code);
|         return(OK);
|     }
|     else
|     {
|         printf("An error occurred while clearing the card");
|
|         printf("card.¥n Return/");
|
|         printf("reason codes %ld/%ld¥n¥n", return_code, reason_code);
|
|         return(ERROR);
|     }
| }
|
|     else
|     {
| printf("An error occurred while getting the random token.¥n");
|
| printf("Return/reason codes ");
|
| printf("%ld/%ld¥n¥n", return_code, reason_code);
|
| return(ERROR);
|     }
| }

```

例: 暗号化コプロセッサを再初期化するための ILE RPG プログラム

暗号化コプロセッサを再初期化するには、必要に応じて以下のプログラム例を変更してください。

注: 法律に関する重要な情報については、292 ページの『コードに関するライセンス情報および特記事項』を参照してください。

提供されているプログラム例を使用する場合には、必要に応じて変更します。セキュリティ上の理由から、IBM では、設定されているデフォルト値をそのまま使用するのではなく、これらのプログラム例を修正して使用することをお勧めします。

```

| D*****
| D* REINIT
| D*
| D* Clear the card (reset to manufactured state).
| D*
| D*
| D* COPYRIGHT 5769-SS1 (C) IBM CORP. 2000, 2000
| D*
| D* This material contains programming source code for your
| D* consideration. These example has not been thoroughly
| D* tested under all conditions. IBM, therefore, cannot
| D* guarantee or imply reliability, serviceability, or function
| D* of these programs. All programs contained herein are
| D* provided to you "AS IS". THE IMPLIED WARRANTIES OF
| D* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
| D* ARE EXPRESSLY DISCLAIMED. IBM provides no program services for
| D* these programs and files.
| D*
| D*
| D* Note: Input format is more fully described in Chapter 2 of
| D*       IBM CCA Basic Services Reference and Guide
| D*       (SC31-8609) publication.
| D*
| D* Parameters:

```

```

D*      char * new time 16 characters
D*
D* Example:
D*   CALL PGM(REINIT)
D*
D* Use these commands to compile this program on the system:
D* CRTRPGMOD MODULE(REINIT) SRCFILE(SAMPLE)
D* CRTPGM  PGM(REINIT) MODULE(REINIT)
D*       BNDSRVPGM(QCCA/CSUACFC)
D*
D* Note: Authority to the CSUACFC service program in the
D*       QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Cryptographic_Facility_Control (CSUACFC)
D*
D*****
D*-----
D* Declare variables for CCA SAPI calls
D*-----
D*      ** Return code
DRETURNCODE      S          9B 0
D*      ** Reason code
DREASONCODE      S          9B 0
D*      ** Exit data length
DEXITDATALEN     S          9B 0
D*      ** Exit data
DEXITDATA        S          4
D*      ** Rule array count
DRULEARRAYCNT   S          9B 0
D*      ** Rule array
DRULEARRAY       S          16
D*      ** Verb data length
DVERBDATALEN    S          9B 0
D*      ** Verb data
DVERBDATA       S          8
D*
D*-----
D* Declares for calculating one's complement
D*-----
DBUFFER          DS
DA1              1      2
DA2              3      4
DA3              5      6
DA4              7      8
D*
DWORKBUFF        DS
DINT4            1      4B 0
DINT2            3      4
D*
D*
D*****
D* Prototype for Cryptographic_Facility_Control (CSUACFC)
D*****
DCSUACFC         PR
DRETCODE         9B 0
DRSNCODE         9B 0
DEXTDTALEN      9B 0
DEXTDTA         4
DRARRAYCT       9B 0
DRARRAY         16
DVRBDTALEN     9B 0
DVRBDTA         8
D*
D*-----
D*      ** Declares for sending messages to the
D*      ** job log using the QMHSNDPM API

```

```

D*-----
DMSG          S          75  DIM(3) CTDATA PERRCD(1)
DMSGLENGTH   S          9B 0 INZ(64)
D             DS
DMSGTEXT     S          1    80
DFAILRETC    S          41   44
DFAILRSNC    S          46   49
DMESSAGEID   S          7    INZ(' ')
DMESSAGEFILE S          21   INZ(' ')
DMSGKEY      S          4    INZ(' ')
DMSGTYPE     S          10   INZ('*INFO ')
DSTACKENTRY  S          10   INZ('* ')
DSTACKCOUNTER S         9B 0 INZ(2)
DERRCODE     DS
DBYTESIN     S          1    4B 0 INZ(0)
DBYTESOUT    S          5    8B 0 INZ(0)
C*
C*****
C* START OF PROGRAM *
C* *
C* *
C*-----*
C* Set the keyword in the rule array *
C*-----*
C          MOVE    'ADAPTER1'  RULEARRAY
C          MOVE    'RQ-TOKEN'  RULEARRAY
C          Z-ADD   2           RULEARRAYCNT
C*-----*
C* Set the verb data length to 8 *
C*-----*
C          Z-ADD   8           VERBDATALEN
C*****
C* Call Cryptographic Facility Control SAPI *
C*****
C          CALLP   CSUACFC      (RETURNCODE:
C                               REASONCODE:
C                               EXITDATALEN:
C                               EXITDATA:
C                               RULEARRAYCNT:
C                               RULEARRAY:
C                               VERBDATALEN:
C                               VERBDATA)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      4
C* *-----*
C*          * Send error message *
C* *-----*
C          MOVE    MSG(1)      MSGTEXT
C          MOVE    RETURNCODE  FAILRETC
C          MOVE    REASONCODE  FAILRSNC
C          EXSR    SNDMSG
C          RETURN
C          ENDIF
C*
C* *-----*
C*          * Send success message for the 1st step *
C* *-----*
C          MOVE    MSG(2)      MSGTEXT
C          EXSR    SNDMSG
C*
C*-----*
C* Set the keyword in the rule array for 2nd step *
C*-----*
C          MOVE    'RQ-REINT'  RULEARRAY
C*

```

*/

```

C*-----*
C* Convert the token into the one's complement of it *
C*-----*
C          MOVE      VERBDATA      BUFFER
C          Z-ADD     0              INT4
C          MOVE      A1             INT2
C          EVAL      INT4 = 65535 - INT4
C          MOVE      INT2           A1
C          MOVE      A2             INT2
C          EVAL      INT4 = 65535 - INT4
C          MOVE      INT2           A2
C          MOVE      A3             INT2
C          EVAL      INT4 = 65535 - INT4
C          MOVE      INT2           A3
C          MOVE      A4             INT2
C          EVAL      INT4 = 65535 - INT4
C          MOVE      INT2           A4
C          MOVE      BUFFER         VERBDATA
C*
C*****
C* Call Cryptographic Facility Control SAPI */
C*****
C          CALLP     CSUACFC        (RETURNCODE:
C                                REASONCODE:
C                                EXITDATALEN:
C                                EXITDATA:
C                                RULEARRAYCNT:
C                                RULEARRAY:
C                                VERBDATALEN:
C                                VERBDATA)
C*-----*
C* Check the return code *
C*-----*
C          RETURNCODE  IFGT      4
C*          *-----*
C*          * Send error message *
C*          *-----*
C          MOVE      MSG(1)        MSGTEXT
C          MOVE      RETURNCODE    FAILRETC
C          MOVE      REASONCODE    FAILRSNC
C          EXSR      SNDMSG
C*
C          ELSE
C*          *-----*
C*          * Send success message *
C*          *-----*
C          MOVE      MSG(3)        MSGTEXT
C          EXSR      SNDMSG
C*
C          ENDIF
C          SETON
C*
C*****
C* Subroutine to send a message
C*****
C          SNDMSG      BEGSR
C          CALL        'QMHSNDPM'
C          PARM        MESSAGEID
C          PARM        MESSAGEFILE
C          PARM        MSGTEXT
C          PARM        MSGLENGTH
C          PARM        MSGTYPE
C          PARM        STACKENTRY
C          PARM        STACKCOUNTER
C          PARM        MSGKEY
C          PARM        ERRCODE
C          ENDSR

```

```
| **  
| CSUACFC failed with return/reason codes 9999/9999.  
| Random token was successfully returned.  
| The Cryptographic Coprocessor successfully cleared/reset.
```

| ハードウェア・サービス管理プログラムの使用

| ハードウェア保守管理機能は、論理的な観点およびパッケージングの観点から、システムのハードウェアを
| 表示および操作するためのツールであり、入出力 (I/O) プロセッサおよびデバイスのデバッグを支援する
| ものです。また、暗号化コプロセッサを再初期化するためにも使用されます (初期化されていない状態に
| 戻します)。

| 暗号化コプロセッサが再初期化されると、暗号化コプロセッサのライセンス内部コードがコプロセッサ
| ーに再ロードされます。コプロセッサのライセンス内部コード用のプログラム一時修正 (PTF) の一部
| (すべてではない) では、PTF を活動状態にするために、ハードウェア保守管理機能の使用を要求すること
| があります。ライセンス内部コードのいくつかのセグメントを再ロードすると、マスター鍵、保管 RSA 秘
| 密鍵、役割、およびプロファイルを含む構成データが失われることから、回復の準備をできるようにするた
| め、この追加のステップが組み込まれています。

| 暗号化コプロセッサを初期化されていない状態にリセットしなければならない場合もあります。例えば、
| コプロセッサの構成が正しくない場合には、コプロセッサの便利な機能をまったく実行できなかった
| り、暗号化コプロセッサの構成ユーティリティーやユーザー作成のアプリケーションを使って訂正できな
| かったりします。もう 1 つの例は、管理プロファイル用のパスワードを忘れてしまい、さらにパスワード
| の変更を許可されている役割を使用するプロファイルが他にない場合です。

| ハードウェア保守管理機能は、システム保守ツールに含まれています。ハードウェア・サービス・マネー
| ジャーを使用する手順は、次のとおりです。

- | 1. システム保守ツール開始 (STRSST) CL コマンドを使用します。CL コマンド行で STRSST と入力し、
| Enter キーを押します。「保守ツールの開始 (STRSST) のサインオン」画面が表示されます。

保守ツールの開始 (STRSST) のサインオン

システム : SYSTEM01

選択項目を入力して、実行キーを押してください。

保守ツール・ユーザー ID _____
保守ツール・パスワード _____

F3= 終了 F9= パスワード変更 F12= 取り消し

- | 2. 保守ツールのユーザー・プロファイル名とパスワードを入力します。「システム保守ツール」画面が表
| 示されます。

システム保守ツール (SST)

次の 1 つを選択してください。

1. 保守ツールの開始
2. 活動保守ツールの処理
3. ディスク装置の処理
4. ディスケット・データ回復の処理
5. システム区画の処理
6. システム容量の処理

選択項目

1

F3= 終了 F10= コマンド入力 F12= 取り消し

3. 保守ツールを開始するには、**1** を選択して、Enter キーを押してください。「保守ツールの開始」画面が表示されます。

保守ツールの開始

警告: この保守ツールを正しく使用しないと、このシステムのデータが損傷することがあります。援助が必要な場合には、弊社技術員に連絡してください。次の 1 つを選択してください。

1. プロダクト活動ログ
2. ライセンス内部コードの追跡
3. 通信追跡の処理
4. 表示/変更/ダンプ
5. ライセンス内部コード・ログ
6. 主記憶域ダンプ管理機能
7. ハードウェア保守管理機能

選択項目

7

F3= 終了 F12= 取り消し F16= SST メニュー

4. ハードウェア保守管理機能を開始するには、**7** を選択してください。ハードウェア保守管理機能の画面に、使用可能なオプションのメニューが表示されます。

ハードウェア保守管理機能

注意: このユーティリティーは、技術員専用提供されています。

システム装置 : 9406-270 10-4314M

リリース : V5R1M0 (1)

次の 1 つを選択してください。

1. パッケージ・ハードウェア資源 (システム、フレーム、カード、...)
2. 論理ハードウェア資源 (バス、IOP、制御装置、...)
3. 資源名による資源の検索
4. 障害があり非報告のハードウェア資源
5. システム電源制御ネットワーク (SPCN)
6. サービス処置ログの処理
7. ラベルのロケーション・ワーク・シートの表示
8. 装置並行保守

選択項目

2

F3= 終了 F6= 構成の印刷 F9= カード・ギャップ情報の表示

F10= アテンションが必要な資源の表示 F12= 取り消し

5. 論理ハードウェア資源を使用するには 2 を選択してください。

論理ハードウェア資源

次の 1 つを選択してください。

1. システム・バス資源
2. プロセッサ資源
3. 主記憶域資源
4. 高速リンク資源

選択項目

1

F3= 終了 F6= 構成の印刷 F12= 取り消し

6. システム・バス資源を表示するには、「論理ハードウェア資源」画面から、1 を選択してください。

システム・バス上の論理ハードウェア資源

処理するシステム・バス *ALL *ALL, *SPD, *PCI, 1-511
 サブセット *CRP *ALL, *STG, *WS, *CMN, *CRP

オプションを入力して、Enter キーを押してください。
 2= 詳細の変更 4= 除去 5= 詳細の表示 6=I/O デバッグ
 7= システム情報の表示
 8= パッケージ資源関連 9=IOP に関連した資源

OPT	記述	タイプ-型式	状況	資源名
-	HSL 入出力ブリッジ	2249-	操作可能	BC02
-	バス拡張アダプター	-	操作可能	BCC02
-	システム・バス	2249-	操作可能	LB01
-	マルチアダプター・ブリッジ	2249-	操作可能	PCI01D
-	多重機能 IOP	* < 284D-001	操作可能	CMB01
-	HSL 入出力ブリッジ	283B-	操作可能	BC01
-	バス拡張アダプター	-	操作可能	BCC03

続く ...

F3= 終了 F5= 最新表示 F6= 印刷 F8= 非報告資源の組み込み
 F9= 障害資源 F10= 非報告資源
 F11= 製造番号/部品番号の表示 F12= 取り消し

7. 暗号化コプロセッサが含まれている IOP が分かっている場合には、IOP の次に **9** を入力します。不明の場合には、「サブセット」フィールドに *CRP と入力してリストをサブセットにし、次に暗号化コプロセッサを含む IOP の隣に **9** を入力します。「IOP に関連した論理ハードウェア資源」画面が表示されます。

IOP に関連した論理ハードウェア資源

オプションを入力して、Enter キーを押してください。
 2= 詳細の変更 4= 除去 5= 詳細の表示 6=I/O デバッグ
 7= 検査 8= パッケージ資源関連

OPT	記述	タイプ-型式	状況	資源名
-	多重機能 IOP	* < 284D-001	操作可能	CMB01
-	暗号アダプター	4758-023	操作可能	CRPCTL01
6	暗号装置	4758-023	操作可能	CRP01
-	ワークステーション IOA	2746-001	操作可能	CTL01
-	表示装置	3477-0FC	操作可能	DSP001
-	表示装置	3477-0FC	操作可能	DSP002
-	通信 IOA	2745-001	操作可能	LIN01
-	通信ポート	2745-001	操作可能	CMN01
-	通信ポート	2745-001	操作可能	CMN02
-	通信 IOA	2744-001	操作可能	LIN03
-	通信ポート	2744-001	操作可能	CMN03

続く ...

F3= 終了 F5= 最新表示 F6= 印刷 F8= 非報告資源の組み込み
 F9= 障害資源 F10= 非報告資源
 F11= 製造番号/部品番号の表示 F12= 取り消し

8. 再初期化する暗号装置の隣に **6** を入力し、Enter キーを押してください。

暗号デバッグ機能の選択

次の 1 つを選択してください。

1. フラッシュ・メモリーの再初期設定
2. IOP デバッグ機能の選択

選択項目

1

F3= 終了 F12= 取り消し

9. フラッシュ・メモリーを再初期化する (暗号化コプロセッサのライセンス内部コードを再ロードするには **1** を選択してください。確認のための画面が表示されます。PTF を適用する場合は、必ず、暗号化されたデータと鍵に関して必要な予防措置を取り、マスター鍵をバックアップしてください。Enter キーを押して続行してください。

フラッシュ・メモリーの再初期設定機能

警告:

この暗号装置フラッシュメモリーの再初期設定を実行すると装置上に保管されたすべての鍵情報が破棄されます。そのため、この装置を使用するすべての暗号化されたデータは、使用できなくなります。

注意:

この暗号装置フラッシュ・メモリーの再初期設定は **10 分** かかります。

続行するためには、実行キーを押してください。

F3= 終了 F12= 取り消し

- 再初期化の状況を知らせる以下のような画面が表示されます。この画面は、再初期化が完了するまで更新されます。

フラッシュ・メモリーの再初期設定状況

フラッシュ・メモリーの再初期設定が進行中

見積り時間 : 10.0 分

経過時間 : 2.5 分

再初期化が完了すると、次のようなメッセージが表示されます。

暗号デバッグ機能の選択

次の 1 つを選択してください。

1. フラッシュ・メモリーの再初期設定
2. IOP デバッグ機能の選択

選択項目

-

F3= 終了 F12= 取り消し
暗号装置の再初期設定が正常に完了した。

再初期化が完了した後、必要に応じて各画面で機能キー F3 を押して、システム保守ツールを終了します。

関連概念

275 ページの『暗号化コプロセッサの再初期化』

暗号化コプロセッサを誤ってセットアップした場合には、構成は使用不可になり、すべての暗号機能が実行できなくなります。また、API を使用して回復することもできません。例えば、マスター鍵の設定が許可された役割がなかったり、新規の役割またはプロファイルの変更または作成が許可された役割がない構成となってしまう場合があります。Cryptographic_Facility_Control (CSUACFC) SAPI を使用すると、カードを再初期化するためのハードウェア・コマンドを呼び出すことができます。

2058 暗号化アクセラレーター

2058 暗号化アクセラレーターは、V5R2 (以降) のシステムで使用可能です。

2058 暗号化アクセラレーターは、暗号化コプロセッサほどのハイ・セキュリティは必要としないものの、ホスト・プロセッサの負荷を軽減するために、ハードウェアによる高速化が提供する、パフォーマンスの高い暗号化を必要とするユーザーに対し、競争力のあるオプションを提供します。2058 暗号化アクセラレーターは、鍵のセキュアな保管場所を必要としない SSL アプリケーションのパフォーマンスを向上させるように設計されています。

暗号化サービス API の使用時に 2058 暗号化アクセラレーターを使用して、DES、Triple DES、SHA-1、および RSA の暗号化方式の処理をオフロードすることも可能です。詳しくは、『暗号化サービス API (Cryptographic Services APIs)』を参照してください。

2058 暗号化アクセラレーターでは、暗号化コプロセッサ・ハードウェアのような、改ざんされにくい鍵の保管場所は提供されません。ご使用のシステムのモデルに応じて、最大 8 つの暗号化アクセラレーターをインストールすることが可能です。1 パーティションごとに最大 4 つの暗号化アクセラレーターを取り付けることが可能です。

2058 暗号化アクセラレーターは、RSA 暗号化 (モジュラー指数) 用に最適化された、最大 2048 ビットのデータ鍵長を持つ、特殊なハードウェアを提供します。また、DES、TDES、および SHA-1 の暗号化方式のための関数を提供します。2058 アクセラレーターは、複数の RSA (Rivest, Shamir、および Adleman アルゴリズム) エンジンを使用します。

関連情報

iSeries のパフォーマンス (iSeries Performance)

フィーチャー

2058 暗号化アクセラレーターの機能については、ここに記載された情報を参照してください。

2058 暗号化アクセラレーターのフィーチャーには、以下のようなものがあります。

- 単一カードのハイパフォーマンス暗号化アダプター (標準 PCI カード)
- RSA 暗号化用の設計および最適化
- オンボード・ハードウェア・ベースの RNG (乱数発生ルーチン)
- IBM UltraCypher 暗号エンジンを 5 つ搭載

シナリオ: システム SSL パフォーマンスの向上

このシナリオで会社が注文し、インストールする 2058 暗号化アクセラレーターは、PCI (Peripheral Component Interconnect) カードです。このカードは、SSL/TLS セッションの確立の際に必要な、極めて計算中心の処理を高速化するように、特別に設計されています。このシナリオでは、システムの SSL パフォーマンスが向上するようにカードを構成するためのステップを指定します。

関連タスク

290 ページの『2058 暗号化アクセラレーターの構成』

装置記述を作成して、i5/OS の SSL が RSA 暗号操作を 2058 暗号化アクセラレーターに送るようにしなければなりません。装置記述は、装置記述の作成 (暗号) (CRTDEVCRP) を使用して作成できます。

状況:

ある会社のシステムが、1 日当たり数千のセキュア・インターネット・トランザクションを処理しています。その会社のトランザクションでは、SSL (Secure Sockets Layer) および Transport Layer Security のプロトコル (SSL および TLS、インターネット・トランザクションを保護するための一般的な方式) を使用しています。この会社のシステム管理者である Sue は、追加のアプリケーション・プロセスを行うた

最初に (さらに多くの SSL トランザクションをサポートできるようにすることも含む)、サーバー資源を解放したいと考えています。Sue は、以下の目標を達成できるソリューションを探しています。

- アプリケーション・プロセス (追加の SSL トランザクションも含めて) で使用可能なサーバー資源のサイズを大幅に増やすことができる
- インストールおよび構成の作業が最小限に抑えられる
- 資源管理の必要が最低限に抑えられる

これらの目標に基づいて、Sue は IBM 2058 e-business 暗号化アクセラレーターを注文し、インストールします。(これ以後は、2058 暗号化アクセラレーターと記述します。) 2058 暗号化アクセラレーターは PCI カードで、SSL/TLS セッションの確立の際に必要な、極めて計算中心の処理を高速化するように、特別に設計されています。ハードウェア・フィーチャー・コード 4805 を注文すると、IBM 2058 暗号化アクセラレーターが入手できます。

詳細:

1. システムに 2058 暗号化アクセラレーターがインストールされ、構成されています。
2. システムがネットワークから大量の SSL トランザクション要求を受信します。
3. 2058 暗号化アクセラレーターが、SSL トランザクションの開始時に暗号処理を実行し、SSL トランザクションのデジタル証明書に関連付けられている秘密鍵をキャッシュに入れます。

前提条件および前提事項:

このシナリオでは、Sue が 2058 暗号化アクセラレーターのインストールを計画し、カードを正しく構成したことを前提としています。また、Sue が SSL のデジタル証明書のセットアップをすでに完了していることも前提となっています。

関連概念

290 ページの『2058 暗号化アクセラレーターの計画』
ご使用のサーバーのモデルに応じて、最大 8 つの IBM 暗号化アクセラレーターをインストールすることが可能です。ご使用のサーバーが暗号化アクセラレーターを使用するためのハードウェア要件およびソフトウェア要件を満たしているかどうかを確認する必要があります。

関連タスク

290 ページの『2058 暗号化アクセラレーターの構成』
装置記述を作成して、i5/OS の SSL が RSA 暗号操作を 2058 暗号化アクセラレーターに送るようにしなければなりません。装置記述は、装置記述の作成 (暗号) (CRTDEVCRP) を使用して作成できます。

構成ステップ:

Sue は、会社のシステムの SSL のパフォーマンスを向上させるために、以下のステップを実行します。

1. 2058 暗号化アクセラレーターを提供する、ハードウェア・フィーチャー・コード 4805 を注文します。
2. 2058 暗号化アクセラレーターをインストールします。
3. 2058 暗号化アクセラレーターの装置記述を作成し、装置をオンに変更します。

関連タスク

290 ページの『2058 暗号化アクセラレーターの構成』
装置記述を作成して、i5/OS の SSL が RSA 暗号操作を 2058 暗号化アクセラレーターに送るようにしなければなりません。装置記述は、装置記述の作成 (暗号) (CRTDEVCRP) を使用して作成できます。

2058 暗号化アクセラレーターの計画

ご使用のサーバーのモデルに応じて、最大 8 つの IBM 暗号化アクセラレーターをインストールすることが可能です。ご使用のサーバーが暗号化アクセラレーターを使用するためのハードウェア要件およびソフトウェア要件を満たしているかどうかを確認する必要があります。

ハードウェア要件

IBM e-business 暗号化アクセラレーター (注文可能なフィーチャー・コード 4805、これ以後は 2058 暗号化アクセラレーターと記述します)。4805 フィーチャーは標準 PCI カードであり、以下のサーバー・モデルでサポートされます。

- eServer i5 520、550、570、および 595
- eServer i5 270、810、820、825、830、840、870、および 890
- eServer i5 拡張装置 5074、5075、5078、5079、5088、5094、5095、5294、および 5790

i5/OS および SSL の要件

2058 暗号化アクセラレーターには、OS/400 V5R2M0 (バージョン 5 リリース 2 モディフィケーション 0) のソフトウェアまたは後継の i5/OS ソフトウェアが必要です。

注: V5R3M0 が稼働しているシステムの場合、SSL も使用するソフトウェアで暗号機能を使用可能にするには、Cryptographic Access Provider 128-bit (5722-AC3) ライセンス・プログラム製品もインストールされていなければなりません。

2058 暗号化アクセラレーターの構成

装置記述を作成して、i5/OS の SSL が RSA 暗号操作を 2058 暗号化アクセラレーターに送るようにしなければなりません。装置記述は、装置記述の作成 (暗号) (CRTDEVCRP) を使用して作成できます。

CL コマンドを使用して装置記述を作成するには、以下のステップを実行します。

1. コマンド行に CRTDEVCRP と入力します。
2. プロンプトが表示されたときに、装置の名前を指定します。
3. PKA 鍵ストアのデフォルト名 *NONE を受け入れます。
4. DES 鍵ストアのデフォルト名 *NONE を受け入れます。
5. *NONE の APPTYPE を指定します。
6. 任意: プロンプトが表示されたら、記述を指定します。
7. 装置記述の作成が完了した後、構成変更 (VRYCFG) または構成状況処理 (WRKCFGSTS) CL コマンドを使用して、装置の構成を変更します。

ソフトウェアで生成され、ソフトウェアに保管されるデジタル証明書では、装置がオンになった後で、i5/OS の SSL が自動的に 2058 暗号化アクセラレーターの使用を開始します。SSL および TLS のセッション確立に関連した秘密鍵の処理が 2058 暗号化アクセラレーターへオフロードされます。装置がオフになっている場合は、i5/OS の SSL が、SSL および TLS のセッション確立のための暗号化を、ソフトウェア・ベースの暗号化にスイッチバックします。したがって、秘密鍵の処理がサーバーに戻されます。

注: これは、暗号化コプロセッサ以外で作成された証明書および秘密鍵についてのみ当てはまります。証明書が、暗号化コプロセッサを使用して生成されたものである場合は、その特定の証明書を使用する SSL または TLS のセッションには、暗号化コプロセッサを使用しなければなりません。

関連概念

288 ページの『シナリオ: システム SSL パフォーマンスの向上』




このシナリオで会社が注文し、インストールする 2058 暗号化アクセラレーターは、PCI (Peripheral Component Interconnect) カードです。このカードは、SSL/TLS セッションの確立の際に必要な、極めて計算中心の処理を高速化するように、特別に設計されています。このシナリオでは、システムの SSL パフォーマンスが向上するようにカードを構成するためのステップを指定します。

暗号化ハードウェアの関連情報

製品マニュアルおよび IBM Redbooks™ (PDF フォーマット)、Web サイト、および暗号化ハードウェアのトピックに関連する Information Center トピックについては、ここに記載された情報を参照してください。PDF を表示し、印刷することもできます。

以下の資料で、暗号の概念およびハードウェアに関する追加情報を提供しています。

IBM の資料


- 「IBM 暗号化ハードウェア (IBM Cryptographic hardware)」 (<http://www.ibm.com/security/cryptocards>) には、4758 暗号化コプロセッサ・ハードウェア・ソリューションに関する情報が記載されています。
- 「IBM PCI 暗号化コプロセッサ・ドキュメンテーション・ライブラリー (IBM PCI Cryptographic Coprocessor documentation library)」 (<http://www.ibm.com/security/cryptocards/library.shtml>) には、4764 暗号化コプロセッサの「CCA 3.2x 基本サービス・マニュアル (CCA 3.2x Basic Services Manual)」、および 4758 暗号化コプロセッサの「2.5x CCA 基本サービス・マニュアル (2.5x CCA Basic Services Manual)」があります。これらのダウンロード可能 PDF 文書は、システム分析者およびアプリケーション分析者、また、CCA プログラムの評価または作成に携わるアプリケーション・プログラマーを対象とした資料です。
- 「CCA 基本サービス・マニュアル (CCA Basic Services Manual)」は、IBM Common Cryptographic Architecture (CCA) をサポートするプログラムを評価または作成する、システム分析者、アプリケーション分析者、およびアプリケーション・プログラマーを対象とした資料です。
- 「IBM 暗号化コプロセッサ・ライブラリー (IBM Cryptographic Coprocessor Library)」 は、このマニュアルのダウンロード可能な PDF です。

PDF ファイルの保管

表示用または印刷用の PDF ファイルをワークステーションに保存するには、次のようにします。

1. ブラウザーで PDF を開く (上記のリンクをクリックする)。
2. PDF をローカルに保管するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「保存」をクリックする。

Adobe Reader のダウンロード

PDF ファイルを表示したり印刷したりするには、Adobe Reader が必要です。これは、Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) からダウンロードできます。

コードに関するライセンス情報および特記事項

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、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。

IBM、そのプログラム開発者、または供給者は、いかなる場合においてもその予見の有無を問わず、以下に対する責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

付録. 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。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。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を提供し、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 Web 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 Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

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、IBM 機械コードのご使用条件、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。

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認できません。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をお願いします。

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。これらの名称はすべて架空のものであり、名称や住所が類似する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。

- | 強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラ
- | ム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、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。

- | IBM、そのプログラム開発者、または供給者は、いかなる場合においてもその予見の有無を問わず、以下に
- | 対する責任を負いません。

- | 1. データの喪失、または損傷。
- | 2. 特別損害、付随的損害、間接損害、または経済上の結果的損害
- | 3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

- | 国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

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的創作物にも、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_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

以下は、IBM Corporation の商標です。

DB2
developerWorks
IBM
iSeries
Operating System/400
OS/400
pSeries
Redbooks
xSeries
zSeries

他の会社名、製品名およびサービス名などはそれぞれ各社の商標または登録商標です。

資料に関するご使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態を提供され、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なしで提供されます。



Printed in Japan