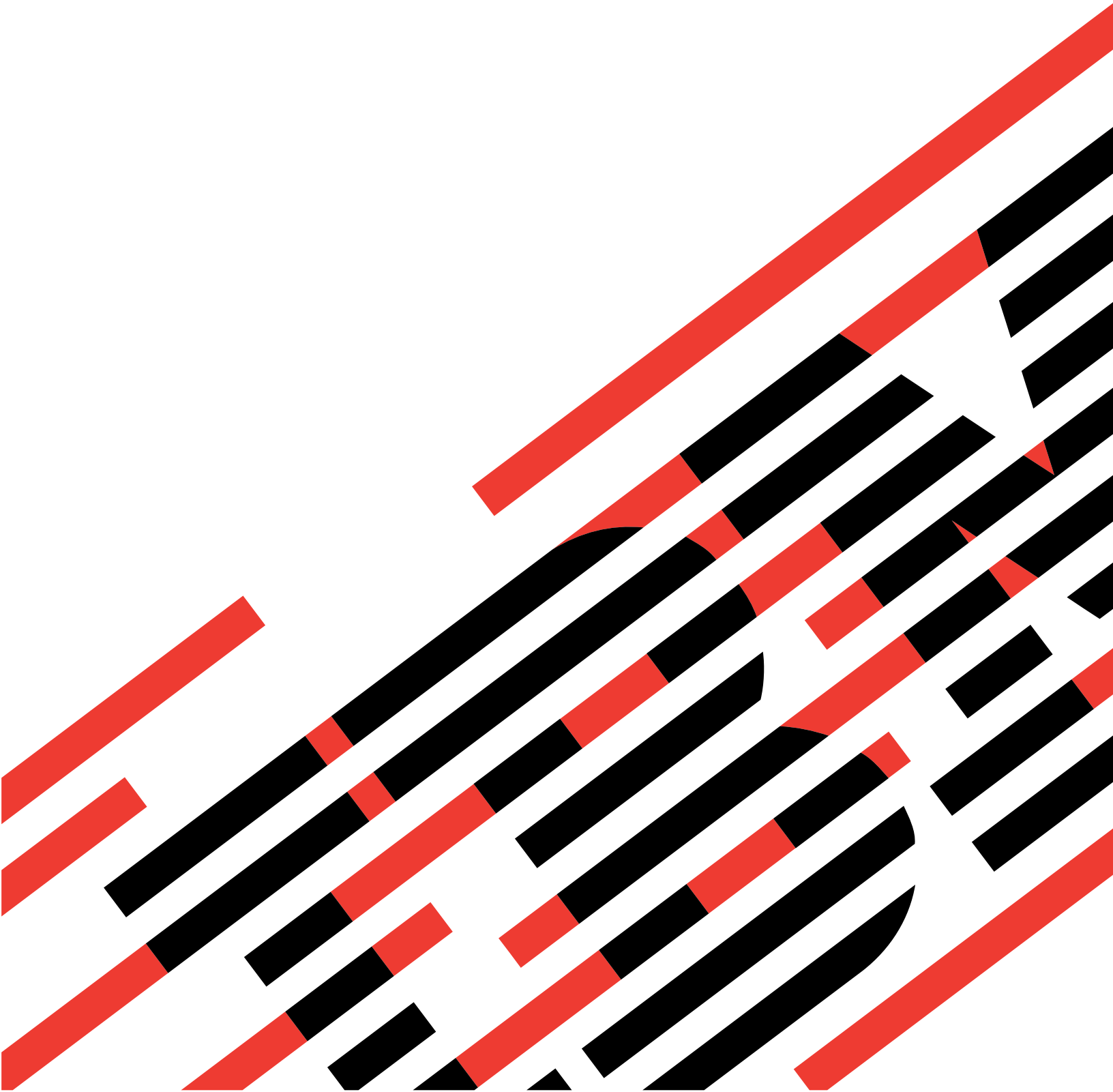
IBM

eserver

iSeries

暗号化ハードウェア



IBM

@server

iSeries

暗号化ハードウェア

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)

原　典： RZAJ–C000–04
iSeries
Cryptographic hardware

発　行： 日本アイ・ビー・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 2002.11

目次

第 1 部 4758 暗号化コプロセッサー

第 1 章 トピックの印刷

この文書の 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』(約 1322 KB、298 ページ)。ここには、iSeries™ V5R2 サーバー用にサポートされている、IBM® 暗号化ハードウェアに関するすべての情報が記載されています。
- 『暗号化ハードウェア: 4758 暗号化コプロセッサー』(約 1317 KB、304 ページ)。ここには、iSeries V5R2 サーバー用にサポートされている、4758 暗号化コプロセッサー・ハードウェアに関する情報が記載されています。

PDF ファイルの保管

表示用または印刷用の PDF ファイルを Netscape Navigator から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 を右マウス・ボタンでクリックする。
2. 「**リンクを名前を付けて保存**」(Netscape Navigator) または「**対象をファイルに保存**」(Internet Explorer) 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 Acrobat 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 Acrobat Reader が必要です。これは、Adobe Web サイト  からダウンロードできます。

第 2 章 V5R2 の新機能

新しい暗号化ハードウェアに関する最新情報、および iSeries サーバー用の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。

»

新しい暗号化ハードウェア: IBM 2058 e-business 暗号化アクセラレーター

4758 暗号化コプロセッサーに加えて、IBM 2058 e-business 暗号化アクセラレーター (ハードウェア・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) が使用可能です。秘密鍵の処理をシステム・プロセッサーの外部に転送することで、iSeries のパフォーマンスを向上させる設計となっており、大量の SSL (Secure Sockets Layer) トランザクションを処理するため、iSeries のインプリメンテーションにおいて、このハードウェア・オプションは優れた選択肢です。2058 暗号化アクセラレーターは、iSeries サーバーの SSL パフォーマンスを向上させるための優れた選択肢ではありますが、4758 コプロセッサーが提供するような広範囲の構成オプションは提供しません。

追加機能: 4758 暗号化コプロセッサー

4758 暗号化コプロセッサーは、以下の新しい機能を提供します。

- 金融サービスにおける PIN 処理: トランザクションごとの固有キー (UKPT)
- Common Cryptographic Architecture (CCA) 2.4

新しい暗号化ハードウェアのシナリオ

暗号化ハードウェアを iSeries サーバーでどのように使用できるかについて例示するために、以下のシナリオを iSeries Information Center に追加しました。

- 『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: 4758 暗号化コプロセッサーを使用する OS/400® アプリケーションの作成』

このリリースでの新機能または変更点についてのその他の情報は、「プログラム資料説明書」

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。

- » のイメージは、新しい情報または変更された情報の始まりを示しています。

-  のイメージは、新しい情報または変更された情報の終わりを示しています。

第 3 章 概念

暗号

暗号は、データを安全に保持するための技術です。基本的な暗号化サービスにより、メッセージがプライベートなものであること、メッセージの保全性が保持されること、通信している当事者が認証されていること、および通信に関与している一方の当事者がメッセージの送信を拒否できないことが保証されます。

暗号を使用すると、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を理解不能なデータ部分 (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。両方のプロセスには、数学の公式またはアルゴリズム、および秘密データ (キー) がかかわっています。

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、1 つのキーが、通信する 2 つの当事者間で共用秘密鍵になります。暗号化および暗号化解除の両方に、同じキーを使用します。秘密鍵アルゴリズムの例としては、データ暗号化規格 (DES) および Triple-DES があります。
2. 公開鍵アルゴリズム、つまり**非対称鍵アルゴリズム**では、キーのペアが使用されます。一方のキーである秘密鍵は秘密にされ、誰とも共用されません。もう一方のキーである公開鍵は秘密ではなく、他人と共用されます。これらのキー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キー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 つのキー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。

どちらのタイプのアルゴリズムでも、データの変更方法を判断するためにキーを使用します。さまざまな暗号プロセスは、いくつかの目的のうちの 1 つを達成するために 1 つのアルゴリズムを使用しています。メッセージ確認コード (MAC) を生成してデータの保全性を保証する場合のように、目的によって使用する暗号化プロセスを選択します。 4758 暗号化コプロセッサー用のユーザー作成アプリケーションは、対応するセキュリティー・アプリケーション・プログラミング・インターフェース (SAPI) を使用して暗号プロセスを呼び出します。キーと暗号プロセスが一緒になってデータを変換します。 SAPI の権限を持つユーザーは、その暗号プロセスにアクセスすることができます。したがって、キー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、キーを保護する必要があります。キー値を秘密にしておけば、そのキーでアルゴリズムを使用することにより、データのセキュリティーを保証することができます。

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。ユーザー・アプリケーションは、キーの生成、選択および分配を完全に制御します。ユーザー・アプリケーションは、暗号化するデータおよび非暗号化テキストとして保管するデータも制御します。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。4758 コプロセッサーは、フィールド・レベルの暗号化と、SSL (Secure Sockets Layer) のセッション確立のための暗号化の両方をサポートしていますが、VPN および SNA のセッション・レベルでの暗号化はサポートしていません。2058 暗号化アクセラレーターは、SSL のセッション確立のための暗号化のみをサポートしています。

データ保全性

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。これは、データ認証性およびデータ保全性と呼ばれます。 4758 コプロセッサーは、メッセージ確認コード (MAC)、メッセージ要約、あるいはディジタル署名を作成することにより、認証性と保全性を保証しています。

メッセージ確認コード (MAC)

MAC プロセスは、重要なデータ要素を定義するデータ保全のための技法です。たとえば、資金転送メッセージで金額を定義することができます。 MAC は、重要なデータ要素、暗号アルゴリズム、および機密 MAC キーから構成されます。 MAC は、メッセージの一部になり、メッセージと共に送信されます。MAC プロセスは DES キーまたは Triple-DES キーを使用します。

メッセージの受信側は、送信側と同じ MAC キー、アルゴリズム、およびプロシージャーを使用して MAC を複製します。受信側の MAC がメッセージと一緒に送られた MAC と一致すると、変更されていないものとして MAC を受け入れることができます。

MAC プロセスでは、受信したメッセージを認証できるようになりますが、転送データは非暗号化テキストのままなので許可なしに読み取られます。 MAC プロセスを使用し、さらにメッセージ全体を暗号化すると、データのプライバシーと保全性を効率的に保護することができます。

メッセージ要約

メッセージ要約のプロセスをデータに対して実行すると、暗号的に生成されたチェックサムと見なされる要約値を作成することができます。データの一部が変更された場合は、異なる要約が生成されます。メッセージ要約のコピーを保持しておき、それらを比較することができます。メッセージ要約が等しいということは、データが変更されていないことを示します。

ディジタル署名

ディジタル署名も、認証性と保全性を検証するために使用できます。これは、以下の 2 ステップから成るプロセスです。

1. 最初にダイジェストがデータから生成され、次にそのダイジェストが RSA 秘密鍵を使用して暗号化されます。この結果がディジタル署名になります。ディジタル署名は、公開鍵を使用してその署名を暗号解除し、オリジナルのダイジェストを回復することにより検証することができます。
2. もう 1 つのダイジェストは、データから生成され、オリジナルのダイジェストと比較されます。この 2 つが同一であれば、その署名は証明されたことになり、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。

4758 暗号化コプロセッサーに関連するキーのタイプ

4758 コプロセッサーはさまざまなキー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 キーあるいは Triple-DES キー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) キーを使用できるわけではありません。4758 コプロセッサーが使用する各種キー・タイプのリストを以下に示します。

マスター・キー

このキーはクリア・キーで、他のキーが暗号化されていないことを意味します。4758 コプロセッサーは、マスター・キーを使用してすべての操作キーを暗号化します。4758 コプロセッサーは、マスター・キー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・キーは、4758 コプロセッサーからは取り出すことはできません。4758 コプロセッサーは、改ざんに対しては、マスター・キーとその工場認証を破棄することにより対応します。4758-023 は、DES キーの暗号化用と PKA キーの暗号化用にそれぞれ 1 つずつの、2 つのマスター・キーを持っています。

倍長鍵暗号鍵

4758 コプロセッサーは、このタイプの Triple-DES キーを使用して他の DES キーまたは Triple-DES キーを暗号化、または暗号化解除します。鍵暗号鍵は、通常、システム間でキー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、オフラインでキーを保管するために使用することもできます。鍵暗号鍵をキーの転送に使用する場合は、鍵暗号鍵自体のクリア値を 2 つのシステム間で共用しなければなりません。エクスポーターの鍵暗号鍵は、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・キーで暗号化されたキーは、暗号化解除され、次に鍵暗号鍵で暗号化されます。インポーターの鍵暗号鍵は、インポート操作に使用されます。インポート操作では、鍵暗号鍵で暗号化されたキーは、暗号化解除され、次にマスター・キーで暗号化されます。

倍長 PIN キー

4758 コプロセッサーはこのタイプのキーを使用して、金融操作で使用される PIN を生成、検証、暗号化、および暗号化解除を行います。これらのキーは Triple-DES キーです。

MAC キー

4758 コプロセッサーは、このタイプのキーを使用して、メッセージ確認コード (MAC) を生成します。これらのキーは、DES キーまたは Triple-DES キーのいずれかです。

暗号鍵 4758 コプロセッサーは、このタイプのキーを使用して、データの暗号化または暗号解除を行います。これらのキーは、DES キーまたは Triple-DES キーのいずれかです。

単一長互換性キー
4758 コプロセッサーは、このタイプのキーを使用してデータの暗号化または暗号化解除を行い、さらに MAC を生成します。これらのキーは DES キーであり、 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 4758 コプロセッサーは、ディジタル証明書の生成、および公開鍵で暗号化された DES キーまたは Triple-DES キーの暗号化解除に秘密鍵を使用します。

公開鍵 4758 コプロセッサーは、ディジタル署名の妥当性検査、DES キーまたは Triple-DES キーの暗号化、および秘密鍵で暗号化されたデータの暗号化解除に公開鍵を使用します。

キー形式

4758 コプロセッサーは、4 つの異なる形式のうちいずれか 1 つで、キーを操作します。キー形式は、キー・タイプと一緒になって、暗号プロセスがそのキー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
キーのクリア値は、どの暗号手段によっても保護されません。クリア・キーは、4758 コプロセッサーでは使用できません。クリア・キーは、最初にセキュア・モジュールにインポートして、マスター・キーで暗号化し、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・キーで暗号化されたキーの形式は、操作可能形式です。これらのキーは、4758 コプロセッサーでは暗号操作に直接使用できます。操作キーは、内部キーとも呼ばれます。サーバーの鍵ストア・ファイルに保管されているキーはすべて操作キーです。しかし、すべての操作キーを鍵ストア・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鍵で暗号化されたキーの形式は、エクスポート形式です。これらのキーは外部キーとも呼ばれます。エクスポート形式のキーは、エクスポーターの鍵暗号鍵と同じクリア・キー値を持つインポーターの鍵暗号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。キーは、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鍵で暗号化されたキーの形式は、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、インポート形式のキーのみです。これらのキーは外部キーとも呼ばれます。インポート形式のキーは、インポーターの鍵暗号鍵と同じクリ

ア・キー値を持つエクスポーターの鍵暗号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。キーは、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、ディ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、4758 コプロセッサー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。機能制御ベクトルは、システムに導入される IBM Cryptographic Access Provider (5769-ACx) 製品に付属しています。ファイルのパス名は、/QIBM/ProdData/CAP/FCV.CRT です。機能制御ベクトルを使用すると、4758 コプロセッサーにキーの作成に必要なキー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、以下を制御するキーに関連付けられた既知の値です。

- キーの型
- このキーが暗号化できる他のキーの種類
- ユーザーの 4758 コプロセッサーがこのキーをエクスポートできるかどうか
- このキー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キーにリンクしているため、制御ベクトルを変更する場合は、同時にキーの値も変更しなければなりません。

鍵ストア・ファイル

4758 コプロセッサーのマスター・キーで暗号化されたキーを保管するために使用される、OS/400 データベース・ファイルです。

キー・トークン

暗号キー、制御ベクトルおよびキー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。キー・トークンは、キーに作用する、あるいはキー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。

第 4 章 4758 暗号化コプロセッサー

iSeries サーバーでは、以下の方法で 4758 コプロセッサーを使用することができます。

- 4758-023 コプロセッサーを DCM と共に使用すると、SSL ディ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。また 4758-023 コプロセッサーでは、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図られています。
- 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、iSeries サーバーでは、複数の 4758-023 コプロセッサー (最大 8 つ) を SSL で使用することができます。複数のコプロセッサーを使用する場合、DCM 構成では、ハードウェアを使用してディ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ーから出て行くことはないため、その秘密鍵を使用したり、他のコプロセッサーと共用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ー間で共用することができます。そのための要件として、各コプロセッサーは、同じマスター・キーを共用しなければなりません。 198 ページの『マスター・キーの複製』機能を使用して、同じマスター・キーを使用するコプロセッサーをセットアップすることができます。秘密鍵は、コプロセッサーの 1 つで生成され、次に鍵ストア・ファイルに保管され、そのコプロセッサーのマスター・キーで暗号化されます。同一のマスター・キーを持っていれば、どのコプロセッサーでもその秘密鍵を使用することができます。
- OS/400 アプリケーションのインプリメントに、4758-023 コプロセッサーが使用できます。これを行うには、システム管理者またはアプリケーション・プロバイダーが CCA CSP API を使用して、4758 コプロセッサーにアクセスするためのアプリケーション・プログラムを作成しなければなりません。そのようなアプリケーションの例として、金融サービスにおける PIN 処理トランザクション、銀行から手形交換所へのトランザクション、および基本的な SET™ ブロック処理があります。CCA CSP を介して最大 8 つのコプロセッサーを使用できます。アプリケーションは、CCA API の Cryptographic_Resource_Allocate (CSUACRA) と Cryptographic_Resource_Deallocate (CSUACRD) を使用して、個々のコプロセッサーへのアクセスを制御しなければなりません。

4758 コプロセッサーについての詳細は、以下のページを参照してください。

14 ページの『4758 暗号化コプロセッサーのフィーチャー』

4758 コプロセッサーには、SSL 中および OS/400 アプリケーションで使用される、暗号操作用のハードウェア・エンジンが含まれています。

Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP) は、OS/400 オプション 35 としてパッケージされています。4758 コプロセッサーの暗号化サービスへのアクセスが可能なアプリケーションを書き込む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) を利用することができます。

フィーチャーのページは、4758 コプロセッサーと CCA CSP で利用できるものについて詳細に説明しています。

20 ページの『4758 暗号化コプロセッサーの要件』

4758 コプロセッサー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、4758 コプロセッサーをサーバーに導入して使用できるかどうかを判断してください。

7 ページの『第 3 章 概念』

暗号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。

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。

4758 暗号化コプロセッサーのフィーチャー

4758 PCI 暗号化コプロセッサーは暗号処理機能を備えており、暗号鍵を安全に保管することができます。サポートされている暗号機能には、データの機密を保持するための暗号化と暗号化解除、データが変更されていないことを保証するためのメッセージ要約とメッセージ確認コード、ディジタル証明書の生成と検証、および金融サービスにおける PIN 処理と SET 処理があります。コプロセッサーは、OS/400 SSL、またはシステム管理者あるいはアプリケーション・プロバイダーによって作成された OS/400 アプリケーションで使用することができます。

コプロセッサーを SSL で使用すると、FIPS 140-1 で認証済みのハードウェア・モジュールで、秘密鍵を作成して保管することができるほか、秘密鍵を複数のコプロセッサー・カードで使用できるように、ソフトウェアで秘密鍵を作成し、マスター・キーでそれを暗号化して、保管することができます。マスター・キーは常に、FIPS 140-1 で認証済みのハードウェア・モジュールに保管されます。また、コプロセッサーは、SSL セッションの確立中は、計算中心の暗号処理を iSeries サーバーのメイン CPU からオフロードします。

OS/400 アプリケーションの場合は、Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP) API を使用して、コプロセッサーの暗号機能と鍵管理機能にアクセスします。

4758-023 PCI 暗号化コプロセッサーは、DES、 Triple-DES、 RSA、 MD5、 SHA-1、 RIPEMD-160、および金融サービスにおける PIN サービスをサポートしま

す。さらに、SET ブロック暗号サービスもサポートします。4758 コプロセッサーの主な利点は、暗号キーを保管するための機能を提供する点です。この機能は、セキュア・モジュールと呼ばれる、改ざんに対して対応する、バッテリー・バックアップされたモジュールで行われます。4758-023 PCI 暗号化コプロセッサーは、Federal Information Processing Standard (連邦情報処理標準) (FIPS) の PUB 1400-1、レベル 3 の要件を満たしています。もう 1 つの利点として、4758 コプロセッサーは、SSL セッションの確立中に、iSeries のメイン CPU での計算中心の暗号処理のオフロードを実現できる点が挙げられます。

4758 コプロセッサーは、役割ベースのアクセス制御機能を備えています。この機能を使用すると、コプロセッサーがサポートしている個々の暗号操作にアクセスでき、それを制御することができます。

CCA CSP フィーチャー

4758 コプロセッサーを CCA CSP と一緒に使用すると、アプリケーションにハイレベルの暗号セキュリティーを行うことができます。顧客または第三者のアプリケーションについては、一連のアプリケーション・プログラミング・インターフェース (API) を介してこれらのサービスにアクセスします。これらの API については、「IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide」 を参照してください。これらの API は、OS/400 オプション 35 - Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP) で提供されます。

4758 コプロセッサーは CCA マスター・キーを使用して、4758 コプロセッサーの外部に保管できるようキーを暗号化します。暗号化されたキーは、データベース・ファイルである鍵ストアファイルに保管します。

IBM の CCA を使用すると、主要な IBM コンピューティング・プラットフォーム上で、暗号に対して一貫したアプローチを取ることができます。OS/400 CCA CSP は、ILE C、RPG、および COBOL で作成されたアプリケーション・ソフトウェアをサポートしています。アプリケーション・ソフトウェアは、データ暗号化規格 (DES) および RSA アルゴリズムを含む、広範囲の暗号機能を実行するために、CCA サービスを呼び出すことができます。

OS/400 CCA CSP は、NT、AIX®、および OS/2® で使用できる、CCA サポート・プログラムと同等の API サポートを提供します。CCA CSP は 4758 コプロセッサーの機能を活用して、次のことを行うことができます。

- ユーザーに与えるアクセスのレベルを定義するための役割ベースのアクセス制御の作成。
- 無作為番号の生成。
- マスター・キーの複製の安全な作成。
- 金融サービスにおける PIN 処理のサポート。
- ディジタル署名の生成と妥当性検査。
- データの暗号化と暗号化解除。
- キーの保護。
- 暗号化された DES キーおよび Triple-DES キーの安全なインポートとエクスポート。

- メッセージ確認コード (MAC) の生成。

4758 暗号化コプロセッサー用の暗号化ハードウェアのシナリオ

この暗号化ハードウェアを iSeries サーバーと共にどのように使用できるかという一例を示すために、以下の使用例のシナリオを追加しました。

- 『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』

  このシナリオは、iSeries サーバーの 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、ディジ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。
- 『暗号化ハードウェアのシナリオ: 4758 暗号化コプロセッサーを使用する OS/400 アプリケーションの作成』

  このシナリオは、4758 暗号化コプロセッサーを呼び出して、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、OS/400 プログラマーにとって役立ちます。

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

状況

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サーバーに、iSeries サーバーを使用しています。この会社の iSeries サーバー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らされています。その要件とは、Sam の会社が実行する iSeries サーバーの 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、ディ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 は、IBM 4758-023 暗号化コプロセッサー PCI カード (ハードウェア・フィーチャー・コード 4801 または 4802、これ以後は 4758 暗号化コプロセッサーと記述します) という、iSeries サーバーで使用可能な暗号化ハードウェア・オプションがあり、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および保管できることを知ります。

Sam は 4758 暗号化コプロセッサーについて調べ、このコプロセッサーを OS/400 ディジタル証明書マネージャー (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 のセッション確立時に実行される暗号操作を iSeries サーバーからオフロードすることで iSeries サーバーのパフォーマンスが向上することを知ります。

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、iSeries サーバーの SSL で複数 (最大 8 つ) の 4758 暗号化コプ

ロセッサーを使用することができます。詳細については、 iSeries での複数の暗号化ハードウェア・カードのインプリメントについて参照してください。

Sam は、自社の iSeries サーバーのセキュリティーを強化するという要件を、4758 暗号化コプロセッサーが満たすと判断します。

詳細

1. この会社の iSeries サーバーに 4758 暗号化コプロセッサー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. 4758 暗号化コプロセッサーによって、秘密鍵が生成されます。
3. その後、秘密鍵が 4758 暗号化コプロセッサーに保管されます。
4. 4758 暗号化コプロセッサーが、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. iSeries サーバーに 4758 暗号化コプロセッサーがインストールされ、正しく構成されていることが前提です (『4758 暗号化コプロセッサーの計画』および『4758 暗号化コプロセッサーの構成』を参照)。4758 暗号化コプロセッサーの計画には、iSeries サーバーで 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 ハンドシェーク処理および秘密鍵の保護のために複数の 4758 暗号化コプロセッサー・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 の会社ではディ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. 4758 暗号化コプロセッサーは、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。

構成ステップ

iSeries サーバー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 ディジタル証明書マネージャー (DCM) を使用して、新しいディジタル証明書を作成するか、あるいは現在のディ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、新しいディ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 4758 暗号化コプロセッサーに保管されます。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。«

暗号化ハードウェアのシナリオ: 4758 暗号化コプロセッサーを使用する OS/400 アプリケーションの作成

»

状況

あなたが、大規模な金融関連の Credit Union 社の、iSeries サーバーのプログラマーであるとします。そして、Credit Union 社の iSeries サーバーにインストールされている IBM 4758-023 暗号化コプロセッサー PCI カード (ハードウェア・フィーチャー・コード 4801 または 4802、これ以後は 4758 暗号化コプロセッサーと記述します) 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されました。

4758 コプロセッサー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、iSeries サーバー OS/400 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。4758 暗号化コプロセッサー用に作成された iSeries サーバー OS/400 アプリケーション・プログラムは、コプロセッサーを使用して、セキュリティーを重視するタスクおよび暗号操作を実行します。

注: CCA CSP を介して最大 8 台の 4758 暗号化コプロセッサーを使用できます。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ー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 iSeries サーバーに送信されます。
3. iSeries サーバー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 4758 暗号化コプロセッサーに送信します。
5. 4758 暗号化コプロセッサーが、PIN の妥当性を確認または拒否します。
6. プログラムが、4758 暗号化コプロセッサーからの結果を 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 iSeries サーバーがあり、そこに 4758 暗号化コプロセッサー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. 『4758 暗号化コプロセッサーの計画』
   b. 『4758 暗号化コプロセッサーの構成』
   c. 『OS/400 アプリケーションで使用するための 4758 暗号化コプロセッサーの構成』
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 OS/400 オプション 35 としてパッケージされており、4758 暗号化コプロセッサー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』 にアクセスできる必要があります。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

4758 暗号化コプロセッサーを使用して 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、2 つの iSeries サーバー OS/400 アプリケーションを作成する方法があります。

1. PIN 検査キーと PIN 暗号キーの両方をロードし、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・キー・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)
   - オプショナル 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。

関連ページ: 『OS/400 アプリケーションで使用するための 4758 暗号化コプロセッサーの構成』

4758 暗号化コプロセッサーの計画

以下の情報は、iSeries サーバーでの 4758 暗号化コプロセッサー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。

- 『4758 暗号化コプロセッサーの要件』
- 『4758 暗号化コプロセッサーへのセキュア・アクセス』
- 『SAPI に必要なオブジェクト権限』

4758 暗号化コプロセッサーの要件

4758 コプロセッサーを導入して使用する前に、サーバーは以下の要件を満たす必要があります。

ハードウェア要件

iSeries サーバー用の 4758-023 コプロセッサーは、フィーチャー・コード 4801 または 4802 を指定することにより注文できます。4801 フィーチャーは、以下の iSeries サーバー・モデルでサポートされます。

- 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
- SB2 および SB3
- 拡張装置 5074、5075、5078、5079、5088、5094、5095 および 5294

4758 コプロセッサーは PCI カードです。コプロセッサーに同梱されているインストール・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。

注: 4758 コプロセッサーは、温度が -15 ℃ より低くなると工場認証が無効となります。4758 コプロセッサーの工場認証が無効になると、カードは使えなくなります。新しいカードの注文については、ハードウェア・サービス提供元に問い合わせてください。

ソフトウェア要件

4758 コプロセッサーには、以下のソフトウェアが必要です。

- OS/400 (5722-SS1): 4758-023 コプロセッサー (iSeries サーバーのフィーチャー・コード 4801 または 4802) には、OS/400 バージョン 4 リリース 5 モディフィケーション 0 以降が必要です。
- OS/400 オプション 35 Common Cryptographic Architecture Cryptographic Service Provider (CCA CSP)
- OS/400 オプション 34 ディジタル証明書マネージャー (4758 コプロセッサー構成のための Web ベースのユーティリティーを使用する予定の場合)
- OS/400 57xx-TC1 TCP/IP 接続ユーティリティー (4758 コプロセッサー構成のための Web ベースのユーティリティーを使用する予定の場合)
- OS/400 57xx-DG1 IBM HTTP Server (4758 コプロセッサー構成のための Web ベースのユーティリティーを使用する予定の場合)
- 4758 暗号化コプロセッサーの暗号化機能を使用可能にするには、Cryptographic Access Provider 128-bit for iSeries (5722-AC3) ライセンス・プログラム製品が、iSeries サーバーにインストールされていなければなりません。このオプションを

使用すると、4758 コプロセッサーで 56 ビット DES キー、112 ビット Triple-DES キー、および 2048 ビット RSA キーを使用することができます。

注:

1. 4758 コプロセッサー構成のための Web ベースのユーティリティーは、OS/400 バージョン 5 リリース 2 モディフィケーション 0 の新機能です。
2. OS/400 バージョン 4 リリース 5 モディフィケーション 0 上で 4758-023 コプロセッサーを SSL と共に使用する場合は、その前に使用制限つきの専用の PTF をインストールしなければなりません。OS/400 バージョン 5 リリース 2 モディフィケーション 0 または OS/400 のそれ以降のリリースで 4758-023 コプロセッサーを SSL と共に使用する場合は、専用の PTF は必要ありません。
3. 以前のバージョンの Cryptographic Access Provider (たとえば、5769-AC1、5769-AC2、5769-AC3 など) がインストールされている場合があります。これらの製品は、バージョン 5 リリース 2 モディフィケーション 0 と互換性があります。

4758 暗号化コプロセッサーへのセキュア・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。ユーザーの組織は、組織のセキュリティー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。

OS/400 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。

4758 コプロセッサー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、4758 コプロセッサー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。 22 ページの『SAPI に必要なオブジェクト権限』を参照してください。

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 4758 管理者の役割を割り当てることを考慮してください。このようにすると、*ALLOBJ 特殊権限を持っているユーザーは、4758 上の管理役割にログオンできないため、コプロセッサーの構成を変更することはできません。しかし、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、4758 管理者による誤用を防ぐことができます。

4758 暗号化コプロセッサー構成のための 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。

4758 コプロセッサーは、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。 4758 コプロセッサー・アクセス制御を使用すると、4758 コプロセッサー・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。これらのコマンドについての詳細は、 27 ページの『役割およびプロファイルの作成と定義』を参照してください。

セキュリティーをさらに強化するには、4758 内部の省略時の役割の機能を制限します。マスター・キーの変更のような、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、 27 ページの『役割およびプロファイルの作成と定義』で説明されている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ー手段を考慮する必要があります。

SAPI に必要なオブジェクト権限

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[4]	Y		Y[2]	Y			
CSNBCPA	Y	Y[1]		Y[1]			
CSNBCPE	Y	Y[1]		Y[1]			
CSNBCSG[4]	Y	Y[1]		Y[1]			
CSNBCSV[4]	Y	Y[1]		Y[1]			
CSNBCVE[4]	Y	Y[1]		Y[1]			
CSNBCVG[4]							
CSNBCVT[4]	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]			
CSNBKEX	Y	Y[1]		Y[1]			
CSNBKGN	Y	Y[2]	Y[2]	Y[1]			
CSNBKPI	Y		Y[1]	Y[1]			
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]

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)
CSNBKTC	Y		Y[1]	Y[1]			
CSNBKTP[4]							
CSNBKTR	Y	Y[1]		Y[1]			
CSNBKYT	Y	Y[1]		Y[1]			
CSNBMDG[4]	Y						
CSNBMGN	Y	Y[1]		Y[1]			
CSNBMKP	Y						
CSNBOWH							
CSNBPEX[4]	Y	Y[1]		Y[1]			
CSNBPGN	Y	Y[1]		Y[1]			
CSNBPTR	Y	Y[1]		Y[1]			
CSNBPVR	Y	Y[1]		Y[1]			
CSNBTRW[4]	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[4]							
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]
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						

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)
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、4758-023 コプロセッサーにのみ関係します。

4758 暗号化コプロセッサーの構成

4758 コプロセッサーを構成すると、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 4758 コプロセッサーを構成するには、4758 暗号化コプロセッサー構成のための 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「iSeries server 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ーの構成 (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。

4758 コプロセッサーがすでに構成済みの場合は、「**構成の管理 (Manage configuration)**」オプションをクリックして、コプロセッサー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、コプロセッサー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 (CSNBMKP)、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ー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。

4758 暗号化コプロセッサー構成ユーティリティーを使用するか、独自にアプリケーションを作成するかに関係なく、4758 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。

1. 『装置記述の作成』。装置記述は、省略時のキーの保管場所を指定します。装置記述は、鍵ストアファイルあり、または、なしで作成することができます。
2. 26 ページの『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キー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。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、装置記述上に名前を付けることもできます。 1 つのファイルに名前を付けてデータ暗号化規格 (DES) キーと Triple-DES キーを保管し、別のファイルに名前を付けて公開鍵アルゴリズム (PKA) キーを保管することもできます。キーを保管するファイルに名前を付けて、DES キー (および Triple-DES キー) と PKA キーを含むデータベースをセットアップします。キーをユーザー独自の鍵ストア・ファイルに保管したい場合には、プログラムを使用して鍵ストア・ファイルに名前を付けます。プログラムによって鍵ストア・ファイルに名前を付けない場合には、CCA CSP は装置記述上に名前が付けられた鍵ストア・ファイルにキーを保管します。
3. 27 ページの『役割およびプロファイルの作成と定義』。ユーザーにこれらの役割とプロファイルを割り当てる場合には、4758 コプロセッサーがユーザーに使用を許可する暗号機能を決定します。
4. 66 ページの『環境 ID およびクロックの設定』。4758 コプロセッサーは、キー・トークンを作成した 4758 コプロセッサーを検査します。 4758 コプロセッサーは、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。
5. 77 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、キーを作成するために使用するキー長を 4758 コプロセッサーに通知します。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。
6. 89 ページの『マスター・キー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・キーをロードして設定します。マスター・キーを使用すると他のキーを暗号化することができます。

装置記述の作成

4758 コプロセッサー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、4758 コプロセッサー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、4758 コプロセッサーに対して鍵ストア・ファイルの省略時の保管場所を示します。4758 暗号化コプロセッサー構成ユーティリティー (http://*server-name*:2001 の「iSeries server 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「iSeries server Tasks」ページにアクセスします。
2. 「4758 暗号化コプロセッサーの構成」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 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 コマンド行で 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。省略時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。省略時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. プロンプトが表示されたら、記述を指定します。この指定はオプションです。
6. 装置記述の作成が完了したら、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。
7. 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。

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キーを使用して何らかの操作を実行する前に、キー・ファイルに名前を付ける必要があります。名前を付けると、4758 コプロセッサーが正しいファイルを指すようになります。 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) キーと Triple-DES キーを保管します。DES および Triple-DES は対称暗号アルゴリズムであり、4758 コプロセッサーは同じキーを使用して、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) キーを保管します。公開鍵アルゴリズムは非対称で、キーはペアで作成されます。4758 コプロセッサーは、1 つのキーを使用して暗号化し、別のキーを使用して暗号化解除します。4758 コプロセッサーは、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、4758 コプロセッサーは、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。

暗号キー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、暗号キーを安全な形式で保管する必要があります。暗号キーは、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、各タイプのキー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、暗号キー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

4758 コプロセッサーは、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、4758 コプロセッサー・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ハードウェア・コマンドに依存しています。 4758 コプロセッサーを使用すると、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ー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ー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ー・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、4758 コプロセッサー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、4758 コプロセッサー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、セキュリティー・ポリシーの意味についても検討してください。

すべての 4758 コプロセッサーは、省略時の役割と呼ばれる役割を持っていなければなりません。 4758 コプロセッサーにログオンしていないユーザーは、省略時の役割で定義されている機能を操作することになります。省略時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省略時の役割で操作し、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。

4758 コプロセッサーが初期化されていない状態にある場合は、省略時の役割では、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
- 装置の再初期化
- 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
- 公開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省略時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、4758 コプロセッサー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ー上の「事故」を防ぐことができます。

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、4758 暗号化コプロセッサー構成のための 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「iSeries server Task」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、コプロセッサー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、省略時の役割の再定義を行うことができます。4758 コプロセッサーが初期設定済みである場合は、「**構成の管理 (Manage configuration)**」、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。4758 コプロセッサーで省略時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。参考のために、4 つのプログラム例が提供されています。そのうちの 2 つは ILE C で作成されており、残りの 2 つは ILE RPG で作成されています。どちらのセットも実行する機能は同じです。

- 32 ページの『例: 4758 コプロセッサー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』
- 53 ページの『例: 4758 コプロセッサーの省略時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』
- 44 ページの『例: 4758 コプロセッサー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』
- 58 ページの『例: 4758 コプロセッサーの省略時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』

注: 提供されているプログラム例の 1 つ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それを使用することをお勧めします。

プロファイルを定義する

4758 コプロセッサーの役割を作成、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、省略時の役割では使用できない、4758 コプロセッサー特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、4758 暗号化コプロセッサー構成のための 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「iSeries server 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、コプロセッサー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。4758 コプロセッサーが初期化済みである場合は、「**構成の管理 (Manage configuration)**」—>「**プロファイル (Profiles)**」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。以下の 2 つのプログラム例が、参考として提供されています。

- 61 ページの『例: 4758 コプロセッサー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』
- 63 ページの『例: 4758 コプロセッサー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』

注: 提供されているプログラム例の 1 つ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それを使用することをお勧めします。

SSL に 4758 コプロセッサーを使用する場合は、省略時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。

- ディジタル署名の生成
- ディジタル署名の検査
- PKA キーの生成
- PKA 複製キーの生成
- RSA 暗号化クリア・データ
- RSA 暗号解読クリア・データ
- 保管キーの削除
- 保管キーのリスト

4758 暗号化コプロセッサー構成ユーティリティーの基本構成ウィザードは、変更しなくても省略時の役割を SSL に使用できるよう、省略時の役割を自動的に再定義します。

セキュリティー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。最大で、必要とされる機能のみを特定的に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・キーの最初の部分のロード
- マスター・キー・パーツの結合
- マスター・キーの設定
- ランダム・マスター・キーの生成
- 新規マスター・キー・レジスターのクリア
- 旧マスター・キー・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、4758 コプロセッサー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。
- 装置の再初期化
- アクセス制御システムの初期化
- 認証データの変更 (たとえば、パスフレーズ)
- パスワード失敗カウントのリセット
- パブリック・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・キー複製制御の初期化
- 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 複製キーの生成 (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
- 保管キーの削除 (SSL のために必要なアクセス制御ポイント)
- 保管キーのリスト (SSL のために必要なアクセス制御ポイント)
- マスター・キーの下での暗号化
- データ・キーのエクスポート
- データ・キーのインポート
- マスター・キーへの再暗号化
- マスター・キーからの再暗号化
- 最初のキー・パーツのロード
- キー・パーツの結合
- キー・パーツの追加
- キー・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。 4758 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、4758 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、4758 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。たとえば、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・キー・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・キー・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・キーの最初の部分のロード
- マスター・キー・パーツの結合
- マスター・キーの設定
- ランダム・マスター・キーの生成
- 新規マスター・キー・レジスターのクリア
- 旧マスター・キー・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キー・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。

複製キー・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・キー複製制御の初期化
- 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 複製キーの生成
- 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

4758 コプロセッサー用のプロファイルを作成、定義した後で、 77 ページの『機能制御ベクトルのロード』に説明されているように、4758 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、4758 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: 4758 コプロセッサー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

4758 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/*  CRTROLEPRF                                                        */
/*                                                                    */
/*  Sample program to create roles and profiles in the 4758           */
/*  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 4758 CCA Basic Services Reference and Guide              */
/*       (SC31-8609) publication.                                     */
/*                                                                    */
/* Parameters:                                                        */
/*   none.                                                            */
/*                                                                    */
/* Example:                                                           */
/*   CALL PGM(CRTROLEPRF)                                             */
/*                                                                    */
/* Use these commands to compile this program on iSeries server:      */
/* 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 4758 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 for iSeries           */
#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};
char role3_bitmap2[] =
    { 0xD8, 0x00, 0x00, 0x1F, 0x00, 0x1F, 0xE6, 0x0F };

/*-------------------------------------------------------------*/
/* 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;

/*-----------------------------------------------------------------*/
/* 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;
    short                   lower_time;
    short                   upper_time;
    char                    valid_days_of_week;
    char                    reserved2;
   } role_header;

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

char * verb_data2;


char * work_ptr;
char *bitmap1, *bitmap2;
int i;                      /* Loop counter                        */

/*-----------------------------------------------------------------*/
/* >>>>>>>> Start of code <<<<<<<<<<<<<<<<<                        */
/*-----------------------------------------------------------------*/
/*-----------------------------------------------------------------*/
/* Allocate storage for the aggregate role structure               */
/*-----------------------------------------------------------------*/
verb_data2 = malloc(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);

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 4758.                                     */
 /*-----------------------------------------------------------*/
```

```
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");
}
```

例: 4758 コプロセッサー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

4758 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D*************************************************************
     D* CRTROLEPRF
     D*
     D* Sample program to create 3 roles and 3 profiles in the 4758
     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 4758 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 iSeries:
     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*----------------------------------------------------------------
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
DTOTAUTDTALEN                    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
DVRBDTA1                        278
DVRBDTALEN2                      9B 0
DVRBDTA2                        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
DHSHLEN                          9B 0
DHSH                            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                   MOVEL     ROLESTRUCT    ROLE1
C*    Set ROLE name  (ROLE2)
C                   MOVEL     '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                   MOVEL     ROLESTRUCT    ROLE2
C*    Set ROLE name  (ROLE3)
C                   MOVEL     '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                   MOVEL     ROLESTRUCT    ROLE3
C*----------------------------------------------------------*
C* Set up roles in verb data 1                              *
C*----------------------------------------------------------*
C*    Set Profile name (SECOFR1)
C                   MOVEL     'SECOFR1 '    USERID
C*    Set Role name (ROLE1)
C                   MOVEL     '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                   MOVEL     PROFILESTRUCT PROF1
C*    Set Profile name (SECOFR2)
C                   MOVEL     'SECOFR2 '    USERID
C*    Set Role name (ROLE2)
C                   MOVEL     '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                   MOVEL     PROFILESTRUCT PROF2
C*    Set Profile name (SECOFR3)
C                   MOVEL     '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                                        LR
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*------------------------------------------------------------*
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
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                   MOVEL     MSG(1)        MSGTEXT
```

```
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     'CSUAACI'     SAPI
C                   EXSR      SNDMSG
C*
C                   ELSE
C*     *----------------------*
C*     * Send success message   *
C*     *----------------------*
C                   MOVEL     MSG(3)        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*
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                                        05
     C                   ENDIF
     C*
     C                   ENDSR
**
CSUAACI failed with return/reason codes 9999/9999.
SECOFR1, SECOFR2, and SECOFR3 profiles were successfully created.
The Default role was successfully changed.
```

例: 4758 コプロセッサーの省略時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

4758 コプロセッサー用の省略時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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/*  SETDEFAULT                                                        */
/*                                                                    */
/*  Sample program to authorize the default role to all access        */
/*  control points in the 4758.                                       */
/*                                                                    */
/*                                                                    */
/*  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 4758 CCA Basic Services Reference and Guide              */
/*       (SC31-8609) publication.                                     */
/*                                                                    */
/* Parameters:                                                        */
/*   none.                                                            */
/*                                                                    */
/* Example:                                                           */
/*   CALL PGM(SETDEFAULT)                                             */
/*                                                                    */
/* Use these commands to compile this program on iSeries:             */
/* 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 for iSeries           */
#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 - 0x127                            */
/*--------------------------------------------------------------------*/
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");
}
```

例: 4758 コプロセッサーの省略時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

4758 コプロセッサー用の省略時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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
D************************************************************
D* SETDEFAULT
D*
D* Sample program to authorize the default role to all access
D* control points in the 4758.
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 4758 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 iSeries:
D* CRTRPGMOD MODULE(SETDEFAULT) SRCFILE(SAMPLE)
D* CRTPGM  PGM(SETEID) 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* Verbdata 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 segements 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)
DMSGLENGTH        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                   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      4
     C*     *-----------------------*
     C*     * Send failure 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
**
CSUAACI failed with return/reason codes 9999/9999.
The Default role was successfully set.
```

例: 4758 コプロセッサー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

4758 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/* Change certain fields in a user profile on the 4758                */
/* 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 4758 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 iSeries:           */
/* ADDLIBLE LIB(QCCA)                                               */
/* CRTCMOD MODULE(CHG_PROF) SRCFILE(SAMPLE)                         */
/* CRTPGM  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 for iSeries            */
#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);
    }
}
```

例: 4758 コプロセッサー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

4758 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D************************************************************
     D* CHG_PROF
     D*
     D* Change certain fields in a user profile on the 4758
```

```
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 4758 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 iSeries:
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
DVRBDTALEN1                      9B 0
DVRBDTA1                         8
DVRBDTALEN2                      9B 0
DVRBDTA2                         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)
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)
C*************************************************************
C* START OF PROGRAM                                          *
C*  *
C*-----------------------------------------------------------*
C* Parameter is profile to be changed.                       *
C*-----------------------------------------------------------*
C     *ENTRY        PLIST
C                   PARM                    VERBDATA1
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'    VERBDATA2
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 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
```

環境 ID およびクロックの設定

環境 ID (EID)

4758 コプロセッサーは、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、4758 暗号化コプロセッサー構成のための 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「iSeries 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、コプロセッサー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。4758 コプロセッサーが初期設定済みである場合は、「**構成の管理 (Manage configuration)**」、次に「**属性 (Attributes)**」とクリックし、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。

- 『例: 4758 コプロセッサーに環境 ID を設定するための ILE C プログラム』
- 69 ページの『例: 4758 コプロセッサーに環境 ID を設定するための ILE RPG プログラム』

4758 コプロセッサーは、4758 コプロセッサーが作成するすべての PKA キー・トークンに EID をコピーします。 EID を使用すると、4758 コプロセッサーは、別の 4758 コプロセッサーが作成したキーではなく、そのコプロセッサーが作成したキーを識別することができます。

クロック

4758 コプロセッサーは、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。省略時の時刻は、グリニッジ標準時刻 (GMT) です。その機能により、設定するための省略時役割の機能を削除する前に、4758 コプロセッサー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、4758 暗号化コプロセッサー構成のための 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「iSeries 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、コプロセッサー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。4758 暗号化コプロセッサーが初期設定済みである場合は、「**構成の管理 (Manage configuration)**」、次に「**属性 (Attributes)**」とクリックし、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。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。

- 72 ページの『例: 4758 コプロセッサーにクロックを設定するための ILE C プログラム』
- 74 ページの『例: 4758 コプロセッサーにクロックを設定するための ILE RPG プログラム』

例: 4758 コプロセッサーに環境 ID を設定するための ILE C プログラム

4578 コプロセッサーで環境 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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/* Set the environment ID on the 4758 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 4758 CCA Basic Services Reference and Guide             */
/*       (SC31-8609) publication.                                    */
/*                                                                   */
/* Parameters:                                                       */
/*   none.                                                           */
/*                                                                   */
/* Example:                                                          */
/*   CALL PGM(SETEID)                                                */
/*                                                                   */
/*                                                                   */
/* 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 iSeries:            */
/* ADDLIBLE LIB(QCCA)                                                */
/* CRTCMOD MODULE(SETEID) SRCFILE(SAMPLE)                            */
/* CRTPGM  PGM(SETEID) MODULE(SETEID)                                */
/*         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 for iSeries             */
#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);
    }
}
```

例: 4758 コプロセッサーに環境 ID を設定するための ILE RPG プログラム

4578 コプロセッサーで環境 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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D*************************************************************
     D* SETEID
     D*
     D* Set the environment ID on the 4758 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 4758 CCA Basic Services Reference and Guide
D*       (SC31-8609) publication.
D*
D* Parameters: None
D*
D* Example:
D*   CALL PGM(SETEID)
D*
D* Use these commands to compile this program on iSeries:
D* CRTRPGMOD MODULE(SETEID) SRCFILE(SAMPLE)
D* CRTPGM  PGM(SETEID) MODULE(SETEID)
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)
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*----------------------------------------------------------*
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 Facilty 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 Environment ID was successfully set.
```

例: 4758 コプロセッサーにクロックを設定するための ILE C プログラム

4758 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*-------------------------------------------------------------------*/
/* Set the clock on the 4758 card, based on a string from            */
/* the command line.  The command line string must be of             */
/* form YYYYMMDDHHMMSSWW, 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 4758 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 iSeries:            */
/* ADDLIBLE 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 for iSeries          */
#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 4758 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);
    }
}
```

例: 4758 コプロセッサーにクロックを設定するための ILE RPG プログラム

4758 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D*************************************************************
     D* SETCLOCK
     D*
     D* Set the clock on the 4758 card, based on a string from
     D* the command line.  The command line string must be of
```

```
D* form YYYYMMDDHHMMSSWW, 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 4758 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 iSeries:
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 Facilty 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 4758 card.
It requires a single command line parameter containing the
new date and time in the form YYYYMMDDHHMMSSWW, where WW is the
day of the week, 01 meaning Sunday and 07 meaning Saturday.
```

機能制御ベクトルのロード

27 ページの『役割およびプロファイルの作成と定義』の後に、4758 コプロセッサー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、4758 コプロセッサー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、ディジタル署名済みの値です。 5722-ACx Cryptographic Access Provider 製品の 1 つを導入すると、そのファイルは、 /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、4758 コプロセッサー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、4758 暗号化コプロセッサー構成のための 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「iSeries 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、コプロセッサー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。4758 コプロセッサーが初期設定済みである場合は、「**構成の管理 (Manage configuration)**」、次に「**属性 (Attributes)**」とクリックし、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。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。

- 『例: 4758 コプロセッサー用機能制御ベクトルをロードするための ILE C プログラム』
- 81 ページの『例: 4758 コプロセッサー用機能制御ベクトルをロードするための ILE RPG プログラム』

機能制御ベクトルをクリアする方法を示す例が、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。

- 85 ページの『例: 4758 コプロセッサー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』
- 87 ページの『例: 4758 コプロセッサー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』

4758 コプロセッサー用の機能制御ベクトルをロードすると、 89 ページの『マスター・キーのロードおよび設定』を使用して、キーを暗号化するために使用するマスター・キーをロードして設定することができます。

例: 4758 コプロセッサー用機能制御ベクトルをロードするための ILE C プログラム

4578 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*-------------------------------------------------------------*/
/* Load the Function Control Vector into the 4758 card.       */
/* The Function Control Vector enables the cryptographic      */
/* functions of the 4758 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 4758 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:        */
/* ADDLIBLE 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 for iSeries      */

/*------------------------------------------------------------------*/
/* 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 4758 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 4758 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: 4758 コプロセッサー用機能制御ベクトルをロードするための ILE RPG プログラム

4578 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
D*************************************************************
D* LOAD_FCV
D*
D* Load the Function Control Vector into the 4758 card.
D* The Function Control Vector enables the cryptographic
D* functions of the 4758 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 4758 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 iSeries:
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                        802
DDESCR                 391    470
DFNCCTLVCT             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                         9B 0
DRSNCODE                         9B 0
DEXTDTALEN                       9B 0
DEXTDTA                          4
DRARRAYCT                        9B 0
DRARRAY                         16
DVRBDTALEN                       9B 0
DVRBDTA                        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*------------------------------------------------------------*
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                                        50
C                   DOU       *IN50
C                   EVAL      CHAR = %SUBST(MSGTEXT:INDEX:1)
C     CHAR          IFNE      X'00'
C                   SETON                                         50
C                   ELSE
C                   EVAL      %SUBST(MSGTEXT:INDEX:1) = ' '
C                   SUB       1             INDEX
C     INDEX         IFEQ      0
C                   SETON                                         50
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 Facilty 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: 4758 コプロセッサー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

4578 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*-------------------------------------------------------------*/
/* Clear the Function Control Vector from the 4758 card.       */
/* The Function Control Vector enables the cryptographic       */
/* functions of the 4758 card.  Clearing it from the 4758      */
/* 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 4758 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 4758                  */
```

```
/*-------------------------------------------------------------------*/
  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: 4758 コプロセッサー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

4578 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D*************************************************************
     D* CLEARFCV
     D*
     D* Clear the Function Control Vector from the 4758 card.
     D* The Function Control Vector enables the cryptographic
     D* functions of the 4758 card.  Clearing it from the 4758
     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 4758 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 iSeries:
     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                   MOVEL     '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                                            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 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・キーのロードおよび設定

77 ページの『機能制御ベクトルのロード』の後、マスター・キーをロードし、設定することができます。4758 コプロセッサーは、マスター・キーを使用してすべての操作キーを暗号化します。マスター・キーは、4758 コプロセッサーのセキュア・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鍵です。 4578 コプロセッサーは、マスター・キーを使用して 4758 コプロセッサー外

部にあるこれらのキーを保管できるように他のキーを暗号化します。マスター・キーは、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 ビットのキーです。

マスター・キーのロード

マスター・キーには、新規、現行、および旧という 3 つのレジスターがあります。新規マスター・キー・レジスターは、作成中のマスター・キーを一時的に保持するために使用されます。このマスター・キーは、キーの暗号化には使用されません。現行マスター・キー・レジスターは、新規に生成、インポート、あるいは再暗号化されたキーを暗号化するために現在使用中のマスター・キーを保持します。旧マスター・キー・レジスターは、以前のマスター・キーを保持します。このレジスターに保持されているマスター・キーは、マスター・キーが変更された後にキーを回復するために使用されます。マスター・キーをロードすると、4758 コプロセッサーはそれを、新規マスター・キー・レジスターに配置します。マスター・キーを設定するまで、そこに残っています。

マスター・キーを作成、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 つを使用します。

- 最初のキー・パーツとその後のキー・パーツを別個にロードして全体としてキーの情報を分割しておく。この方法は、安全性が最も低い方法ですが、各キー・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上させることができます。
- ランダムにキーを生成することにより、キーに関する人間の知識を排除する。この方法は、マスター・キーをロ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・キーを 2 番目の 4758 コプロセッサーに複製する必要があります。
- 別のコプロセッサーから複製して、既に存在するマスター・キーを使用する。

マスター・キーの複製についての詳細は、

http://www.ibm.com/security/cryptocards/html/library.shtml  を参照してください。

マスター・キーの設定

マスター・キーを設定すると、現行マスター・キー・レジスター内にあるキーが旧マスター・キー・レジスターに移動します。次に、新規マスター・キー・レジスター内にあるマスター・キーが、現行マスター・キー・レジスターに移動します。

注: マスター・キー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・キーのバックアップ・コピーを取っておくことが不可欠です。たとえば、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、別のコプロセッサーにマスター・キーの複製を作成してください。

最も簡単、かつ迅速にマスター・キーをロードおよび設定するには、4758 暗号化コプロセッサー構成のための 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「iSeries 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、コプロセッサー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。4758 コプロセッサー

が初期設定済みである場合は、「**構成の管理 (Manage configuration)**」、次に「**マスター・キー (Master keys)**」とクリックし、マスター・キーをロードし、設定します。

独自のアプリケーションを作成して、マスター・キーをロードして設定することもできます。それには、Master_Key_Process (CSNBMKP) 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。

- 『例: 4758 コプロセッサーにマスター・キーをロードするための ILE C プログラム』
- 94 ページの『例: 4758 コプロセッサーにマスター・キーをロードするための ILE RPG プログラム』

注: 提供されているプログラム例の 1 つ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それを使用することをお勧めします。

キーの再暗号化

マスター・キーを設定する際、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・キーで暗号化されたすべてのキーを再暗号化する必要があります。再暗号化は、マスター・キーを変更、設定する前に行います。

4758 暗号化コプロセッサー構成のための Web ベースのユーティリティーを使用して、キー保管庫内のキーを再暗号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「iSeries Tasks」ページからアクセスできます。 4758 コプロセッサーは初期設定済みでなければなりません。「構成の管理 (Manage configuration)」、さらに「DES キー」とクリックして DES キーを再暗号化するか、「PKA キー」をクリックして PKA キーを再暗号化します。

キー保管庫にないキーがある場合、あるいはキー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。

- 97 ページの『例: 4758 コプロセッサー用にキーを再暗号化するための ILE C プログラム』

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それを使用することをお勧めします。

例: 4758 コプロセッサーにマスター・キーをロードするための ILE C プログラム

4758 コプロセッサーに新規マスター・キーをロードするには、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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/* Load a new master key on the 4758 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'0123456789ABCDEFFEDCBA98765432100123456789ABCDEF')    */
/*                                                                    */
/* 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 iSeries:             */
/* ADDLIBLE LIB(QCCA)                                                 */
/* CRTCMOD MODULE(LOAD_KM) SRCFILE(SAMPLE)                            */
/* CRTPGM  PGM(LOAD_KM) MODULE(LOAD_KM)                               */
/*         BNDSRVPGM(QCCA/CSNBMKP QCCA/CSNBRNG)                       */
/*                                                                    */
/* Note: Authority to the CSNBMKP and CSNBRNG service programs        */
/*       in the QCCA library is assumed.                              */
/*                                                                    */
/* The main Common Cryptographic Architecture (CCA) verb used         */
/* is Master_Key_Process (CSNBMKP).                                   */
/*                                                                    */
/*--------------------------------------------------------------------*/


#include "csucincl.h"      /* header file for CCA Cryptographic      */
                           /* Service Provider for iSeries            */
#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                                      */
    /*-------------------------------------------------------------------*/
    CSNBMKP( &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);
    }

}
```

例: 4758 コプロセッサーにマスター・キーをロードするための ILE RPG プログラム

4758 コプロセッサーに新規マスター・キーをロードするには、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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D************************************************************
     D* LOAD_KM
     D*
     D* Load a new master key on the 4758 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 4758 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 is 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'0123456789ABCDEFFEDCBA98765432100123456789ABCDEF')
D*
D*   CALL PGM(LOAD_KM)
D*     (MIDDLE X'1032A873458010F7EF3438373132F1F2F4F8B3CDCDCDCEF1')
D*
D*   CALL PGM(LOAD_KM)
D*     (LAST X'2040806789ABCDEFFEDC34343464321001234567 89FEDCBA')
D*
D*   CALL PGM(LOAD_KM) (SET)
D*
D*
D*
D* Use these commands to compile this program on iSeries:
D* CRTRPGMOD MODULE(LOAD_KM) SRCFILE(SAMPLE)
D* CRTPGM  PGM(LOAD_KM) MODULE(LOAD_KM)
D*         BNDSRVPGM(QCCA/CSNBMKP)
D*
D* Note: Authority to the CSNBMKP service program in the
D*       QCCA library is assumed.
D*
D* The Common Cryptographic Architecture (CCA) verbs used are
D* Master_Key_Process (CSNBMKP)
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 CSNBMKP
DKEYPART          S             24    INZ(*ALLX'00')
D*
D************************************************************
D* Prototype for Master_Key_Process (CSNBMKP)
D************************************************************
DCSNBMKP          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     CSNBMKP       (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                                          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*
**
CSNBMKP failed with return/reason codes 9999/9999
The request completed successfully
```

例: 4758 コプロセッサー用にキーを再暗号化するための ILE C プログラム

4758 コプロセッサー用のキー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*-------------------------------------------------------------*/
/* 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 4758 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:               */
/*    ADDLIBLE LIB(QCCA)                                             */
/*    CRTCMOD 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 for iSeries      */

/* 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. */
krl = fopen(data_set_name, "rb");

if (krl == 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,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);

    /* Loop through the entries in the KRL and re-encipher. */
    for (i = 1; i <= num_rec; i++)
    {
 int result;
 result = re_encipher(krl, rec_length, keysto_type);
 if (result !=0)
 {
     fclose(krl);
     return ERROR;
 }
    }
    printf("Key store file re-enciphered successfully.¥n¥n");
    fclose(krl);
    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() */
```

DCM および SSL で使用するための 4758 暗号化コプロセッサーの構成

以下のセクションでは、4758-023 コプロセッサーを 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 4758-023 コプロセッサーを使用する

4758-023 コプロセッサーとその前提条件となるソフトウェアを導入するには、以下の手順を実行する必要があります。

- サーバーにコプロセッサーをインストールします。

  フィーチャー 4801 の場合は、4758-023 コプロセッサーに付属の 4801 PCI 暗号化コプロセッサー・カードの説明書に従って、4758-023 コプロセッサーを導入します。

  フィーチャー 4802 の場合は、IBM ハードウェア・サービス技術員に 4758-023 コプロセッサーのインストールを依頼してください。
- OS/400 オプション 35 CCA CSP をインストールします。
- 5722-AC3 Cryptographic Access Provider (128-bit) ライセンス・プログラム・プロダクトをインストールします。
- 21 ページの『4758 暗号化コプロセッサーへのセキュア・アクセス』 のように OS/400 の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「iSeries Tasks」ページにアクセスします。
- 24 ページの『4758 暗号化コプロセッサーの構成』のステップに従って 4758 を構成します。

以上で、4758-023 コプロセッサー使用して 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。

最後の 2 ステップの詳細については、『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』を参照してください。

注: SSL に複数のカードを使用する場合は、188 ページの『複数の 4758 暗号化コプロセッサーの管理』と 198 ページの『マスター・キーの複製』を参照してください。

OS/400 アプリケーションで使用するための 4758 暗号化コプロセッサーの構成

以下のセクションでは、4758-023 コプロセッサーを OS/400 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。 OS/400 アプリケーションでの 4758 暗号化コプロセッサーの使用例については、『暗号化ハードウェアのシナリオ: 4758 暗号化コプロセッサーを使用する OS/400 アプリケーションの作成』を参照してください。

OS/400 アプリケーションに 4758-023 コプロセッサーを使用する

4758 コプロセッサーとその前提条件となるソフトウェアを導入するには、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ーをインストールします。

  フィーチャー 4801 の場合は、4758 コプロセッサーに付属の 4801 PCI 暗号化コプロセッサー・カードの説明書に従って、4758 コプロセッサーを導入します。

  フィーチャー 4802 の場合は、IBM ハードウェア・サービス技術員に 4758 コプロセッサーの導入を依頼してください。
- OS/400 オプション 35 CCA CSP をインストールします。
- 5722-AC3 Cryptographic Access Provider (128-bit) または 5722-AC2 Cryptographic Access Provider (56-bit) のライセンス・プログラム製品を導入します。
- 21 ページの『4758 暗号化コプロセッサーへのセキュア・アクセス』 のように OS/400 の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「iSeries Tasks」ページにアクセスします。
- 24 ページの『4758 暗号化コプロセッサーの構成』のステップに従って 4758 を構成します。
- 暗号化コプロセッサーを使用するアプリケーションを作成します。

注: OS/400 アプリケーションに複数のカードを使用する場合は、188 ページの『複数の 4758 暗号化コプロセッサーの管理』を参照してください。

4758 暗号化コプロセッサーへの移行

ここでは、以下の移行の実行方法について説明します。

- 『他の iSeries 暗号製品から 4758 暗号化コプロセッサーへの移行』
- 『鍵ストア・ファイルの移行』

他の iSeries 暗号製品から 4758 暗号化コプロセッサーへの移行

以前に暗号の作業を行っている場合、ユーザーのサーバーには次の 2 つの暗号製品のうち 1 つが含まれている可能性があります。すなわち、IBM CCA Services for iSeries (5799-FRF) 製品の鍵ストア・ファイルか、Cryptographic Support for iSeries (5769-CR1) の暗号クロスドメイン・ファイルの、いずれかが存在す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 4758 コプロセッサー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。

- **IBM CCA Services for iSeries (5799-FRF)。**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ハードウェア上で暗号機能を提供します。Common Cryptographic Architecture (CCA) サービスを使用するには、暗号プロセッサー、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、104 ページの『IBM CCA Services for iSeries からの鍵ストア・ファイルの移行』 を使用して、IBM CCA サービスから 4758 コプロセッサーに移行することができます。
- **Cryptographic Support for iSeries (5769-CR1 または 5722-CR1)。**Cryptographic Support は、ホスト・マスター・キーの下にあるクロスドメイン・キー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・キーをファイルに保管します。クロスドメイン・キー・ファイルは、 114 ページの『Cryptographic Support for iSeries のクロスドメイン・キー・ファイルの移行』を使用して、Cryptographic Support for iSeries サーバーから 4758 コプロセッサーに移行することができます。

IBM CCA Services for iSeries からの鍵ストア・ファイルの移行

現在、Common Cryptographic Architecture (CCA) Services for iSeries (5799-FRF) を使用している場合には、4758 コプロセッサーが使用できるように、鍵ストア・ファイル内にあるキーを移行することができます。コプロセッサーは、移行されたキーを CCA Cryptographic Service Provider (OS/400 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 4758 コプロセッサーよりも広範囲のキー・タイプをサポートしているため、すべてのキーを移行することはできません。たとえば、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キーを移行することはできません。さらに、CCA サービスには 4758 コプロセッサー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 for iSeries では、いかなる PKA キーも移行することができません。

データ暗号化規格 (DES) キーを移行するには、2 つのプログラムを作成する必要があります。あるいは、2 つのプログラム例 105 ページの『例: キーのエクスポート』と 110 ページの『例: キーのインポート』を変更して実行しても、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 キー・トークンの形式を定義しているので、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 キーは、IBM CCA Services for iSeries から 4758 コプロセッサーおよび CCA CSP に移行されます。最初に EXPORT プログラムを実行して、必要なキー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・ファイルにキーをファイルからインポートすることができます。キー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。

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. 鍵暗号鍵に対して同じクリア・キー値を両方の製品にインポートします。鍵暗号鍵は、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**各キー**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、出力された外部キー・トークンを CCA CSP および 4758 コプロセッサーにインポートします。**各キー**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。

各キー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: 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それを使用することをお勧めします。

例: キーのエクスポート: 以下は、ステップ 1 です。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。このプログラムを実行した後 110 ページの『例: キーのインポート』を使用して移行プロセスを完了します。

```
/*-------------------------------------------------------------------*/
/* 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 4758.           */
/*                                                               */
/* Note: This program is intended to be used in conjunction with */
/*       IMPORT_TSS to migrate DES keys from 2620 to 4758.       */
/*       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:           */
/*    ADDLIBLE 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 "csucincl.h"        /* header file for CCA Cryptographic
                                Service Provider for iSeries     */
```

```
int main(int argc, char *argv[])
{

   /*-------------------------------------------------*/
   /* standard return codes                           */
   /*-------------------------------------------------*/
#define ERROR -1
#define OK     0

   /*-------------------------------------------------*/
   /* Declare function pointers (see csucincl.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 *krl, *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¥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,'¥x01',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(krl);
   fclose(export_file);
   return OK;
}
```

例: キーのインポート: 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、 105 ページの『例: キーのエクスポート』プログラムを実行して移行プロセスを開始します。次に、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。

```
/*-------------------------------------------------------------*/
/* 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 4758.          */
/*                                                             */
/* Note: This program is intended to be used in conjunction with */
/*       EXPORT_TSS to migrate DES keys from 2620 to 4758.     */
/*       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 4758 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:         */
/*    ADDLIBLE 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 for iSeries     */

/*------------------------------------------------------------*/
/* 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 4758 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 for iSeries のクロスドメイン・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 for iSeries (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・キーは、新しい 4758 コプロセッサーに移行することができます。

Cryptographic Support for iSeries 製品 (5769-CR1 または 5722-CR1) は、ホスト・マスター・キーで自身のクロスドメイン・キーを暗号化して、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、コプロセッサーと共に CCA を使用するために、クロスドメイン・キーを 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・キーによるクロスドメイン・キー暗号化** Cryptographic Support for iSeries は、クロスドメイン・キーのクリア・キー値のインポートをサポートし、さらにクロスドメイン・キーの下でデータ・キーの暗号化をサポートします。ただし、クロスドメイン・キーの下でのクロスドメイン・キーの暗号化はサポートしておらず、クロスドメイン・キーのクリア・キー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・キーの移行は、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。
- **単一長キー対倍長キー** Cryptographic Support for iSeries では、すべてのキーが単一長キーです。 CCA では、すべての鍵暗号鍵および PIN キーが倍長キーです。キーの長さは異なりますが、単一長キーから倍長キーを作成し、その倍長キーを単一長キーのように振る舞わせることができます。倍長キーの両半分が同じ場合、暗号操作の結果は単一長キーを使用した場合と同じになります。したがって、キーを Cryptographic Support for iSeries から CCA に移行する場合は、クロスドメイン・キーのキー値を、CCA キーのキー値フィールドの両半分にコピーする必要があります。
- **CCA 制御ベクトル対マスター・キー変形** CCA では、鍵暗号鍵でキーが暗号化されている、といわれる場合は、実際には、鍵暗号鍵と制御ベクトルの排他論理和演算で得られるキー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・キーは、3 つの異なるマスター・キー変形の 1 つで暗号化されます。マスター・キー変形とは、ホスト・マスター・キーと、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・キー変形はどちらも、キーの分離を提供しているため、キー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、その使用法が決まります。暗号化機能サポートでは、キーの使い方により、そのキーを暗号解読するために使用されるマスター・キー変形が決まります。どちらの場合も、キーをその意図している使い方以外で使用しようとすると、エラーが発生します。制御ベクトルとマスター・キー変形の動作は似ていますが、マスター・キー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。

- **倍長キーに対する CCA 制御ベクトルの非対称** 倍長キーは、その両半分が同じ場合のみ、単一長キーのように動作します。倍長キー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キーは、いかなる場合でも、生成されたキーの両半分は同じにはなりません。この倍長キーは、単一長キーのようには動作しません。

キー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・キー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・キーを、暗号化機能サポートと CCA の間の共通キーとして使用することができます (CCA では、ホスト・マスター・キーは、マスター・キーとして知られています)。暗号化機能サポートのホスト・マスター・キーのクリア値を CCA に IMPORTER 鍵暗号鍵としてインポートします。ホスト・マスター・キーを 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・キーのインポートは、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考えなければなりません。Cryptographic Support のホスト・マスター・キーに対して二重の責務がある場合、この鍵暗号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・キー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キーを 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・キー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・キーをすべてのクロスドメイン・キー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・キー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・キーには次の 3 つのタイプがあります。

- 受信クロスドメイン・キー
- 送信クロスドメイン・キー
- PIN クロスドメイン・キー

受信クロスドメイン・キーは、CCA では IMPORTER 鍵暗号鍵と呼びます。どちらも、暗号化されたキー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・キーは、a) データ・キーの暗号化 (暗号化されたキーは、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キー分割があるため、両機能を実行するキーを生成またはインポートすることはできません。 EXPORTER 鍵暗号鍵および OPINENC (アウトバウンド PIN 暗号化) キーの両方としてキーを使用する場合には、 2 つの異なるキー・タイプを持つ 2 種類のキーに送信クロスドメイン・キーを 2 回インポートする必要があります。

PIN クロスドメイン・キーは、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) キー

に分離します。キーを、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・キーを同様に 2 回インポートする必要があります。

ホスト・マスター・キーがデータ・キーを暗号化するとき、異なるマスター・キー変形がクロスドメイン・キーを暗号化します。

- マスター・キー変形 1 は送信クロスドメイン・キーを暗号化する。バリアント 1 は、16 進数 88 の 8 バイトでホスト・マスター・キーを排他論理和演算した結果です。
- マスター・キー変形 2 は受信クロスドメイン・キーを暗号化する。バリアント 2 は、ホスト・マスター・キーおよび 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・キー変形 3 は PIN クロスドメイン・キーを暗号化する。バリアント 3 は、ホスト・マスター・キーおよび 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・キーのクリア・キー値だけをインポートする場合には、どのキーも移行することができません。キーを移行するには、どのマスター・キー変形でキーを暗号化するのかを考慮する必要があります。

マスター・キー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。キーの移行プロセスは、キー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。IBM 4758 CCA の「IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide」では、こうしたプロセスのための方法を説明しています。この方法は、事前排他論理和演算技法です。キーをインポートする前に鍵暗号鍵 (この場合はホスト・マスター・キー) のクリア・キー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、この鍵暗号鍵がインポートするどのキー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、単一長キーで作業を行っている場合には正しく動作します。倍長キーの場合は、CCA キー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、次のようにキーを 2 回インポートします。

1. 半分の各 8 バイトが、インポートするキー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鍵を作成します。この鍵暗号鍵を使用してインポートされるキーの左半分のみが有効になります。
2. 半分の各 8 バイトが、インポートするキー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鍵を作成します。このインポーター鍵暗号鍵を使用すると、インポートされるキー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キー・トークンとして配置します。

これで、Cryptographic Support for iSeries 製品からクロスドメイン・キーのように働く CCA 倍長キーが作成されました。

130 ページの『IMPORTER 鍵暗号鍵の使用』は、すべてのクロスドメイン・キー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を要約しています。さらに、その鍵暗号鍵の作成方法も示しています。

方法 2

注: この方法は、ご使用のシステムと環境のセキュリティー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・キー・ファイルにとってはセキュリティー・リスクが大きくなります。クロスドメイン・キー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・キーをデータ・キーとして CCA にインポートします。以下に示すように、このキーを、マスター・キー変形に等価なデータ・キー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。
   - マスター・キー変形 1 は送信クロスドメイン・キーを暗号化する。バリアント 1 は、16 進数 88 の 8 バイトでホスト・マスター・キーを排他論理和演算した結果です。
   - マスター・キー変形 2 は受信クロスドメイン・キーを暗号化する。バリアント 2 は、ホスト・マスター・キーおよび 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・キー変形 3 は PIN クロスドメイン・キーを暗号化する。バリアント 3 は、ホスト・マスター・キーおよび 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・キーが作成されます。
2. Decrypt (CSNBDEC) CCA API を使用してクロスドメイン・キーを暗号化解除し、クリア・キー値を戻します。正しいデータ・キー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・キーを 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、すべてのキー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・キーを移行するためのプログラムを作成することも、また、以下のプログラム例を変更することもできます。

例: クロスドメイン・キーを 4758 コプロセッサーに移行し、暗号化機能サポートのクロスドメイン・キーを移行する: Cryptographic Support for iSeries のクロスドメイン・キー・ファイルを 4758 コプロセッサーに移行するには、必要に応じて以下のプログラム例を変更してください。

```
/*********************************************************************/
/* This program migrates keys stored in the file QACRKTBL in library */
/* QUSRSYS to key storage for Option 35 - CCA Cryptographic Service  */
/* Provider.  The QACRKTBL file contains cross domain keys that are  */
/* used for the Cryptographic Support licensed program, 5769-CR1.    */
/*                                                                   */
/*  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.                                        */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/* The keys are migrated by the following steps:                     */
/*                                                                   */
/* 1 - The master key used for 5769-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.  5769-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 4758 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 iSeries:           */
/* ADDLIBLE 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            pingen_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 exlusive 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 varient 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],
                pingen_cv, variant3)               ||

      buildImporter(PINGEN_importerR, argv[1],
                &pingen_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++)              /* Repeat for each record */
       {                               /* Read a record          */
        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.  5769-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 occured 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 occured 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 occured importing an importer key¥n");
           break;
          }
       }
     else
       {
        /* Import the PINGEN key          */
        if(importNonCCA(key_label,
                     PINGEN_importerL,
                     PINGEN_importerR,
                     pingen_cv,
                     key_rec.key_value))
          {
           printf("An error occured 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 occured 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 occured openning 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 5769-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,'¥00',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,'¥00',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;
}
```

IMPORTER 鍵暗号鍵の使用: あらゆるタイプのクロスドメイン・キーをインポートするには、次の IMPORTER 鍵暗号鍵が必要です。

1. エクスポーター・キーの左半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、エクスポーター鍵暗号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・キーの右半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、エクスポーター鍵暗号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・キーの左半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、インポーター鍵暗号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・キーの右半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、インポーター鍵暗号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 キーの左半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、OPINENC キー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 キーの右半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、OPINENC キー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 キーの左半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、IPINENC キー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 キーの右半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、IPINENC キー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 キーの左半分をインポートするための KEK。

   このキーは、クリア・ホスト・マスター・キー、PINGEN キー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。
10. PINGEN キーの右半分をインポートするための KEK。

    このキーは、クリア・ホスト・マスター・キー、PINGEN キー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。
11. PINVER キーの左半分をインポートするための KEK。

    このキーは、クリア・ホスト・マスター・キー、PINVER キー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 キーの右半分をインポートするための KEK。

    このキーは、クリア・ホスト・マスター・キー、PINVER キー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。

鍵ストア・ファイルの移行

ここで示す手順では、Common Cryptographic Architecture Cryptographic Service Provider for iSeries Services から鍵ストア・ファイルを移行するプロセスについて説明します。

4758 暗号化コプロセッサーの管理

このセクションでは主に、4758 コプロセッサーを OS/400 アプリケーションで使用する場合について説明します。複数のコプロセッサーを SSL で使用する場合は、188 ページの『複数の 4758 暗号化コプロセッサーの管理』および 198 ページの『マスター・キーの複製』を参照してください。

4758 コプロセッサーをセットアップした後で、4758 コプロセッサー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。プログラムを使用すると次の作業を実行することができます。

- 役割制限のある API を扱うための『4758 暗号化コプロセッサーのログオンまたはログオフ』
- 142 ページの『状況の照会または情報の要求』
- DES キーおよび PKA キーのレコードを保持する場合の 147 ページの『鍵ストア・ファイルの初期化』
- 153 ページの『DES キーおよび PKA キーの作成』とそれらの DES キー保管庫への保管
- 160 ページの『ファイルの暗号化または暗号化解除』
- 166 ページの『PIN の処理』
- 179 ページの『ディジタル署名の生成および検査』
- 188 ページの『複数の 4758 暗号化コプロセッサーの管理』
- 複数の 4758 コプロセッサーを使用する場合の 198 ページの『マスター・キーの複製』

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それを使用することをお勧めします。

4758 暗号化コプロセッサーのログオンまたはログオフ

ログオン

省略時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。

4758 コプロセッサーにログオンした後で、4758 コプロセッサー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。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。

- 132 ページの『例: 4758 コプロセッサーにログオンするための ILE C プログラム』

- 134 ページの『例: 4758 コプロセッサーにログオンするための ILE RPG プログラム』

ログオフ

4758 コプロセッサーを終了したときは、4758 コプロセッサー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。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。

- 137 ページの『例: 4758 コプロセッサーからログオフするための ILE C プログラム』
- 139 ページの『例: 4758 コプロセッサーからログオフするための ILE RPG プログラム』

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それを使用することをお勧めします。

例: 4758 コプロセッサーにログオンするための ILE C プログラム

4758 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*-------------------------------------------------------------------*/
/* Log on to the 4758 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 4758 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 iSeries:           */
/* ADDLIBLE 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 for iSeries          */
#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 4758 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¥n",
            return_code, reason_code);
    }
    else
      printf("Logon was successful¥n");
}
```

例: 4758 コプロセッサーにログオンするための ILE RPG プログラム

4758 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D*************************************************************
     D* LOGON
     D*
     D* Log on to the 4758 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 4758 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 iSeries:
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
DRSNCODE                         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 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                    USERID
C                  PARM                    AUTHDATA
C*-----------------------------------------------------------*
C* Set the keywords in the rule array                        *
C*-----------------------------------------------------------*
C                  MOVEL     '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                                          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
```

例: 4758 コプロセッサーからログオフするための ILE C プログラム

4758 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/* Log off the 4758 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 4758 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 iSeries:           */
/* ADDLIBLE 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 for iSeries          */
#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 Authenication parm and data lengths must be 0                */
    auth_parm_length = 0;
    auth_data_length = 0;


    /* Invoke verb to log off the 4758 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¥n",
             return_code, reason_code);
     return(ERROR);
    }
    else
    {
     printf("Log off successful¥n");
     return(OK);
    }
}
```

例: 4758 コプロセッサーからログオフするための ILE RPG プログラム

4758 コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D************************************************************
     D* LOGOFF
     D*
     D* Log off from the 4758 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 4758 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 iSeries:
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                        9B 0
DRSNCODE                        9B 0
DEXTDTALEN                      9B 0
DEXTDTA                         4
DRARRAYCT                       9B 0
DRARRAY                        16
DUSR                            8
DATHPRMLEN                      9B 0
DATHPRM                         8
DATHDTALEN                      9B 0
DATHDTA                         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                  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* 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                                            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                                          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キーの長さ、マスター・キーの状況、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、4758 コプロセッサーに照会することができます。最も簡単かつ最も迅速に 4758 コプロセッサーに照会するには、4758 暗号化コプロセッサー構成のための Web ベースのユーティリティーを使用します。「**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、コプロセッサー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。参考のために、2 つのプログラム例が提供されています。 『例: 4758 コプロセッサーの状況を照会する』は、STATEID キーワードと TIMEDATE キーワードを使用します。また一方で、145 ページの『例: 4758 コプロセッサーの情報を要求する』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出されます。

「IBM 4758 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: 4758 コプロセッサーの状況を照会する

4758 コプロセッサー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。

```
/*--------------------------------------------------------------------*/
/* Query the 4758 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 4758 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 iSeries:           */
/* ADDLIBLE 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 for iSeries           */
#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("%ld/%ld¥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("%ld/%ld¥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("%ld/%ld¥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("%ld/%ld¥n¥n", return_code, reason_code);

 return(ERROR);
    }
}
```

例: 4758 コプロセッサーの情報を要求する

4758 コプロセッサー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。

```
/*-------------------------------------------------------------------*/
/* Query the 4758 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 4758 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 iSeries:            */
/* ADDLIBLE 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 for iSeries            */
#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、操作キー、つまりマスター・キーで暗号化されたキーを保管するデータベース・ファイルです。4758 コプロセッサーの 2 つの異なるタイプのキー保管庫を初期化することができます。4758 コプロセッサーは、1 つのタイプを PKA キーを保管するために使用し、もう 1 つのタイプを DES キーを保管する

ために使用します。鍵ストア・ファイルにキーを保管する、あるいはハードウェアに保管されているキーを使用する予定の場合は、鍵ストア・ファイルを初期化する必要があります。

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。

キー保管庫を初期化するために、4758 暗号化コプロセッサー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 (Manage configuration)**」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 キー**」または「**PKA キー**」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) 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。

- 『例: 4758 コプロセッサー用のキー保管庫を初期化するための ILE C プログラム』
- 150 ページの『例: 4758 コプロセッサー用のキー保管庫を初期化するための ILE RPG プログラム』

注: 提供されているプログラム例の 1 つ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それを使用することをお勧めします。

4758 コプロセッサー用のキー保管庫を作成すると、153 ページの『DES キーおよび PKA キーの作成』を使用して DES キーと PKA キーを生成し、鍵ストア・ファイルに保管することができます。

例: 4758 コプロセッサー用のキー保管庫を初期化するための ILE C プログラム

4758 コプロセッサーのキー保管を初期化するには、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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*-------------------------------------------------------------*/
/* 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:            */
/* ADDLIBLE 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 for iSeries      */

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: 4758 コプロセッサー用のキー保管庫を初期化するための ILE RPG プログラム

4758 コプロセッサーのキー保管を初期化するには、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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     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 4758 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 iSeries:
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
DDSCRP                          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
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                    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

4758 コプロセッサーを使用すると 2 つのタイプの暗号キーを作成することができます。

- データ暗号化規格 (DES) キーの内容は、非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キー値を使用します。 160 ページの『ファイルの暗号化または暗号化解除』、 166 ページの『PIN の処理』、およびキーの管理には、DES キーを使用します。

  4758 コプロセッサーで DES キーを作成するには、プログラムを作成するか、または 154 ページの『例: 4758 コプロセッサーで DES キーを作成する』のプログラムを変更します。

- 公開鍵アルゴリズム (PKA) キー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キーを使用します。 179 ページの『ディジタル署名の生成および検査』を使用したファイルの署名、およびキーの管理には、PKA キーを使用します。

  4758 コプロセッサーで PKA キーを作成するには、プログラムを作成するかまたは 156 ページの『例: 4758 コプロセッサーで PKA キーを作成する』のプログラムを変更します。

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それを使用することをお勧めします。

DES キーと PKA キーは、 147 ページの『鍵ストア・ファイルの初期化』を使用して、それらのキー用に作成した鍵ストア・ファイルに保管します。PKA キーは、4758 コプロセッサーに保管することもできます。キーをハードウェアに保管するための詳細については、http://www.ibm.com/security/cryptocards/html/library.shtml の 4758 情報を参照してください。

例: 4758 コプロセッサーで DES キーを作成する

4758 コプロセッサーで DES キーを作成するには、必要に応じて以下のプログラム例を変更してください。

```
/*-----------------------------------------------------------------*/
/* 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 identifed 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:             */
/* ADDLIBLE 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 for iSeries     */

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¥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¥n",
            return_code, reason_code);
    return ERROR;
  }
  else
  {
    printf("Key generated and stored in key store¥n");
    printf("SAPI returned %ld/%ld¥n¥n", return_code, reason_code);
    return OK;
  }
}
```

例: 4758 コプロセッサーで PKA キーを作成する

4758 コプロセッサーで PKA キーを作成するには、必要に応じて以下のプログラム例を変更してください。

```
/*-------------------------------------------------------------*/
/* 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 identifed 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 descrption.                        */
/*                                                             */
/*       This program also assumes the key store file you will */
/*       use is already identifed 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:         */
/* ADDLIBLE LIB(QCCA)                                          */
/* CRTCMOD MODULE(PKAKEYGEN) SRCFILE(SAMPLE)                   */
/* CRTPGM  PGM(PKAKEYGEN) MODULE(PKAKEYGEN)  +                 */
/*         BNDSRVPGM(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 for iSeries     */

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_identifer;
      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 %ld/%ld¥n¥n", return_code, reason_code);
      return OK;
  }
}
```

ファイルの暗号化または暗号化解除

4758 コプロセッサー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するかまたは『例: データを 4578 コプロセッサーで暗号化する』のプログラムを変更して、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 4578 コプロセッサーで暗号化する

4758 コプロセッサー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。

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* 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 identifed 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:              */
/* ADDLIBLE LIB(QCCA)                                               */
/* CRTCMOD MODULE(ENCFILE) SRCFILE(SAMPLE)                          */
/* CRTPGM  PGM(ENCFILE) MODULE(ENCFILE) +                           */
/*         BNDSRVPGM(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 for iSeries     */

/*------------------------------------------------------------------*/
/* 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。4758 コプロセッサーを使用すると、PIN の処理を行うことができます。

PIN を処理するには、プログラムを作成するか、または 『例: 4758 コプロセッサー上での PIN の処理』のプログラムを変更します。

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それを使用することをお勧めします。

例: 4758 コプロセッサー上での PIN の処理

4758 コプロセッサーで 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。

```
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 4758 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 iSeries:
    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 identifed 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 descrption.
    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
DRSNCODE                        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('/')
DRSNCODE                   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                   MOVEL     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                   MOVEL     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                   MOVEL     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                   MOVEL     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*
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 exclusived
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                                          LR
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
C                   SETON                                          LR
C                   ENDIF
C*****************************************************************
C* Import the PINVER key *
C*************************
C                   MOVEL     'FIRST   '    RULEARRAY
C                   Z-ADD     1             RULEARRAYCNT
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 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                   MOVEL     'CSNBKPI'     FAILMESSAGE
C                   EXSR      SNDFAILMSG
C                   SETON                                          LR
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 PINVER key
C*****************************************************************
C                   MOVEL     '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                   MOVEL     'CSNBKPI'     FAILMESSAGE
C                   EXSR      SNDFAILMSG
C                   SETON                                          LR
C                   ENDIF
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 exclusived
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                                          LR
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                                          LR
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
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                   MOVEL     'CSNBKPI'     FAILMESSAGE
C                   EXSR      SNDFAILMSG
C                   SETON                                          LR
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 OPINENC key
C*************************************************************
C                   MOVEL     '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                   MOVEL     'CSNBKPI'     FAILMESSAGE
C                   EXSR      SNDFAILMSG
C                   SETON                                          LR
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                   MOVEL     'IBM-PIN '    RULEARRAY
C                   Z-ADD     8             PINLEN
C                   Z-ADD     8             PINCKL
C                   MOVEL     '01234567'    DECTABLE
C                   MOVE      '89123456'    DECTABLE
C                   MOVEL     '11112222'    VALDATA
C                   MOVE      '33334444'    VALDATA
C*************************************************************
C*  Call Clear PIN Generate
C*************************************************************
C                   CALLP     CSNBPGN       (RETURNCODE:
C                                            REASONCODE:
C                                            EXITDATALEN:
C                                            EXITDATA:
C                                            PINGENKEY:
C                                            RULEARRAYCNT:
C                                            RULEARRAY:
C                                            PINLEN:
C                                            PINCKL:
C                                            DATAARRAY:
C                                            CPIN)
C     RETURNCODE    IFGT      4
C                   MOVEL     'CSNBPGN'     FAILMESSAGE
C                   EXSR      SNDFAILMSG
C                   SETON                                          LR
C                   ENDIF
C*
C*
C*************************************************************
```

```
C* Encrypt the clear PIN using CSNBCPE (Clear_PIN_Encrypt)
C*  Rule_array_count = 1
C*  Rule_array = "ENCRYPT "
C*  PIN Profile = "3624    NONE           F"
C*  PAN data is ignored
C*  Sequence number is ignored but set to 99999 anyway
C*****************************************************************
C                   Z-ADD     1             RULEARRAYCNT
C                   MOVEL     'ENCRYPT '    RULEARRAY
C                   MOVEL     '3624    '    PINFORMAT
C                   MOVE      'NONE    '    FORMATCONTROL
C                   MOVE      '       F'    PADDIGIT
C                   Z-ADD     99999         SEQNUMBER
C*****************************************************************
C*  Call Clear PIN Encrypt
C*****************************************************************
C                   CALLP     CSNBCPE       (RETURNCODE:
C                                            REASONCODE:
C                                            EXITDATALEN:
C                                            EXITDATA:
C                                            OPINKEY:
C                                            RULEARRAYCNT:
C                                            RULEARRAY:
C                                            CPIN:
C                                            PROFILE:
C                                            PAN:
C                                            SEQNUMBER:
C                                            EPIN)
C     RETURNCODE    IFGT      4
C                   MOVEL     'CSNBCPE'     FAILMESSAGE
C                   EXSR      SNDFAILMSG
C                   SETON                                          LR
C                   ENDIF
C*
C*
C*****************************************************************
C* Verify encrypted PIN using CSNBPVR (Encrypted_PIN_Verify)
C*****************************************************************
C                   MOVEL     'IBM-PIN '    RULEARRAY
C
C                   CALLP     CSNBPVR       (RETURNCODE:
C                                            REASONCODE:
C                                            EXITDATALEN:
C                                            EXITDATA:
C                                            IPINKEY:
C                                            PINVERKEY:
C                                            PROFILE:
C                                            PAN:
C                                            EPIN:
C                                            RULEARRAYCNT:
C                                            RULEARRAY:
C                                            PINCKL:
C                                            DATAARRAY)
C     RETURNCODE    IFGT      4
C                   MOVEL     'CSNBPVR'     FAILMESSAGE
C                   EXSR      SNDFAILMSG
C                   SETON                                          LR
C                   ENDIF
C*
C*****************************************************************
C* Send successful completion message
C*****************************************************************
C                   CALL      'QMHSNDPM'
C                   PARM                    MESSAGEID
C                   PARM                    MESSAGEFILE
C                   PARM                    GOODMESSAGE
C                   PARM                    GOODMSGLENGTH
```

```
C                   PARM                    MSGTYPE
C                   PARM                    STACKENTRY
C                   PARM                    STACKCOUNTER
C                   PARM                    MSGKEY
C                   PARM                    ERRCODE
C*
C                   SETON                                           LR
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                    FAILMSGLENGTH
C                   PARM                    MSGTYPE
C                   PARM                    STACKENTRY
C                   PARM                    STACKCOUNTER
C                   PARM                    MSGKEY
C                   PARM                    ERRCODE
C                   ENDSR
```

ディジタル署名の生成および検査

ディジタル署名の生成

ディ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。ディジタル署名には、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ィ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、異なるディ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 キーを使用するには、プログラムを作成するか、または 180 ページの『例: 4758 コプロセッサーによるファイルの署名』のプログラムを変更します。

ディジタル署名の検証

ディ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どうかが分かります。ディ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になります。検証するには、プログラムを作成するかまたは 184 ページの『例: 4758 コプロセッサーによるディジタル署名の検証』を変更します。

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それを使用することをお勧めします。

例: 4758 コプロセッサーによるファイルの署名

4758 コプロセッサー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。

```
/*-------------------------------------------------------------*/
/* 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 4758 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:         */
/*    ADDLIBLE 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 for iSeries      */

/*-------------------------------------------------------------*/
/* 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¥n", h_out);
 return OK;
    }
}
```

例: 4758 コプロセッサーによるディジタル署名の検証

4758 コプロセッサーでディ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。

```
/*-------------------------------------------------------------*/
/* 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 4758 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:         */
/*   ADDLIBLE LIB(QCCA)                                        */
/*   CRTCMOD MODULE(VERFILESIG) SRCFILE(SAMPLE) SYSIFCOPT(*IFSIO)*/
/*   CRTPGM  PGM(SIGNFILE) MODULE(SIGNFILE) +                  */
/*           BNDSRVPGM(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 for iSeries      */

/*-------------------------------------------------------------*/
/* 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 4758 暗号化コプロセッサーの管理

1 つのシステムでは、4758 コプロセッサーを最大 8 つ持つことができます。複数の 4758 コプロセッサー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 つのコプロセッサー (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ーの割り振りを解除して、新しいコプロセッサーを割り振ることにより、コプロセッサー間でジョブを切り替えることができます。 OS/400 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ーを使用するように指定されていれば、コプロセッサー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。

コプロセッサーをすべて同じに構成すると、すべての操作キーはすべてのコプロセッサーで同じように作用します。あるコプロセッサーで暗号化されたデータを、別のコプロセッサーで暗号化解除することができます。すべての鍵ストア・ファイルは、コプロセッサーを交換しても動作します。コプロセッサー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・キーです。あるコプロセッサーに対してマスター・キーをパーツに入力した場合に、そのマスター・キーのパーツを他のコプロセッサー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ーに対しても、同じマスター・キーのパーツを入力しなければなりません。コプロセッサー内でランダムなマスター・キーが生成された場合、すべてのコプロセッサーが区別なく動作するようにするには、他のコプロセッサーにもそのマスター・キーを複製しなければなりません。

すべてのコプロセッサーを同じ構成にしたくない場合もあります。コプロセッサーの構成をすべて異なるものにしたり、コプロセッサー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、すべての操作キーがすべてのコプロセッサーで同じようには働かない可能性があります。あるコプロセッサーで暗号化されたデータが、別のコプロセッサーで回復できないこともあります。また、コプロセッサー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・ファイルと操作キーが、特定のコプロセッサーに対して動作

するのかを追跡しなければなりません。コプロセッサーを異なる構成にすると、暗号アプリケーションのスケーラビリティーが制限されることがありますが、セキュリティーの面では、より高い細分度が提供されます。たとえば、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 キーを使用している場合は、コプロセッサーも交換可能にはなりません。保管キーは、いかなる場合でもコプロセッサー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、その保管キーを使用する暗号要求は、その保管キーを保管しているコプロセッサーに送信しなければなりません。

以下の解説は、OS/400 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。

装置を割り振る

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決定できるように、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、システムは省略時の暗号装置を割り振ります。省略時の装置は、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。省略時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。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。

- 『例: コプロセッサーを割り振るための ILE C プログラム』
- 191 ページの『例: コプロセッサーを割り振るための ILE RPG プログラム』

装置の割り振りを解除する

4758 コプロセッサーを使い終わった後で、Cryptographic_Resource_Deallocate (CSUACRD) API verb を使用して、 4758 コプロセッサー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。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。

- 194 ページの『例: コプロセッサーの割り振りを解除するための ILE C プログラム』
- 196 ページの『例: コプロセッサーの割り振りを解除するための ILE RPG プログラム』

例: コプロセッサーを割り振るための ILE C プログラム

コプロセッサーを割り振るには、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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:            */
/* ADDLIBLE 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);
    }
}
```

例: コプロセッサーを割り振るための ILE RPG プログラム

コプロセッサーを割り振るには、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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
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                                            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*            *----------------------*
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
```

例: コプロセッサーの割り振りを解除するための ILE C プログラム

コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:             */
/* ADDLIBLE 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);
    }
}
```

例: コプロセッサーの割り振りを解除するための ILE RPG プログラム

コプロセッサー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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
     D* CRTRPGMOD MODULE(CRPDEALLOC) SRCFILE(SAMPLE)
     D* CRTPGM  PGM(CRPDEALLOC) MODULE(CRPDEALLOC)
     D*         BNDSRVPGM(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*-------------------------------------------------
     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                  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 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*            *----------------------*
     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*
**
CSUACRD failed with return/reason codes 9999/9999'
The request completed successfully
```

マスター・キーの複製

マスター・キーの複製は、ある 4758 コプロセッサーから別のコプロセッサーに、マスター・キーの値を露出しないで安全にコピーする方法です。これを実行するには、マスター・キーを *n* 個の共用パーツに分割します。 *n* は 1 〜 15 の数です。別のコプロセッサーでマスター・キーを再作成するには、*m* 個の共用パーツが必要になります。ここで、*m* は 1 〜 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・キー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・キーを含むコプロセッサーは、マスター・キー共用ソース・ノード、あるいは送信側と呼ばれます。送信側は、保管 RSA キー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、複製と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、コプロセッサー共用署名キーあるいは送信側キーとして知られています。マスター・キーを受け取るコプロセッサーは、マスター・キー共用ターゲット・ノード、あるいは受信側と呼ばれます。受信側も保管 RSA キーのペアを生成し、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。このキーは、コプロセッサー共用受信キーあるいは単に受信側キー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、コプロセッサー内の保管秘密鍵でディ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。このコプロセッサー内の保管秘密鍵を、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キーです。また、共用管理キーとも呼ばれます。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。4758 コプロセッサーは、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、コプロセッサー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 キーを使って暗号化します。次に triple- DES キー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 のラップが解かれるか、Triple-DES キーを使ってその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・キーが、受信側の新規のマスター・キー・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・キーを複製するには、4758 暗号化コプロセッサー構成のための 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・キー複製アドバイザーが含まれています。マスター・キー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. 「4758 暗号化コプロセッサーの構成 (4758 Cryptographic Coprocessor configuration)」ページの「**構成の管理 (Manage configuration)**」をクリックします。
2. 「**マスター・キー**」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ー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・キー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)

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。

- 201 ページの『例: 4758 コプロセッサーでマスター・キー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』
- 203 ページの『例: 4758 コプロセッサーのマスター・キー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』
- 205 ページの『例: マスター・キーの複製のための保管キーのペアを生成するための ILE C プログラム』
- 210 ページの『例: マスター・キーの複製のための保管キーのペアを生成するための ILE RPG 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 C プログラム』
- 221 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』
- 227 ページの『例: 公開鍵の証明書を登録するための ILE C プログラム』
- 229 ページの『例: 公開鍵の証明書を登録するための ILE RPG プログラム』
- 234 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』
- 238 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』
- 246 ページの『例: マスター・キーの共用パーツを取得するための ILE C プログラム』
- 250 ページの『例: マスター・キーの共用パーツを取得するための ILE RPG プログラム』
- 256 ページの『例: マスター・キーの共用パーツを導入するための ILE C プログラム』
- 260 ページの『例: マスター・キーの共用パーツを導入するための 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・キー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。

- 267 ページの『例: 保管キーをリストするための ILE C プログラム』
- 269 ページの『例: 保管キーをリストするための ILE RPG プログラム』
- 272 ページの『例: 保管キーを削除するための ILE C プログラム』
- 274 ページの『例: 保管キーを削除するための ILE RPG プログラム』

マスター・キーの複製についての詳細は、「IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide」 を参照してください。

例: 4758 コプロセッサーでマスター・キー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

4758 コプロセッサーでマスター・キー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/* Set the M-of-N values in the 4758 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 4758 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 iSeries:            */
/* ADDLIBLE 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 for iSeries             */
#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: 4758 コプロセッサーのマスター・キー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

4758 コプロセッサーでマスター・キー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D*************************************************************
     D* SETMOFN
     D*
     D* Set the M-of-N values in the 4758 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 4758 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 iSeries:
     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_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 contain M (minimum) and N (maximum)
DVERBDATA         DS            8
DM                              9B 0
DN                              9B 0
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                         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)
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                    MVALUE           15 5
C                   PARM                    NVALUE           15 5
C*-----------------------------------------------------------*
C* Set the keyword in the rule array                         *
C*-----------------------------------------------------------*
C                   MOVEL     '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 Facilty 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                   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                   MOVEL     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.
```

例: マスター・キーの複製のための保管キーのペアを生成するための ILE C プログラム

マスター・キーを複製するための保管キー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、295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:            */
/* ADDLIBLE 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 certicate 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 Certficate 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;

/* 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;

/* 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 digtal 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・キーの複製のための保管キーのペアを生成するための ILE RPG プログラム

マスター・キーを複製するための保管キー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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
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
DPRVSECLEN        S              9B 0
D*                ** Public key section length
DPUBSECLEN        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                 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                    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                                            KEYNAMEL:
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                   MOVEL     MSG(1)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     '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        PRVSECLEN
C*    Get the public section length.  The length is at position
C*    (11 + Private key section length).
C                   EVAL      MSB = TOKENARRAY(10 + PRVSECLEN + 1)
C                   EVAL      LSB = TOKENARRAY(11 + PRVSECLEN + 1)
C                   MOVE      LENGTH        PUBSECLEN
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 + PUBSECLEN + 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     PUBSECLEN
C                   EVAL      TOKENARRAY(INDEX + I) =
C                              TOKENARRAY(PRVSECLEN + I + 8 + 1)
C     1             ADD       I             I
C                   ENDDO
C                   EVAL      INDEX  = INDEX + PUBSECLEN - 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 Certficate 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 Certficate 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(PRVSECLEN+PUBSECLEN+12+1))
C                   MOVEL     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                   MOVEL     '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                   MOVEL     MSG(1)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     'CSNDPKG'     SAPI
C                   EXSR      SNDMSG
C                   RETURN
C                   ENDIF
C*
C*      *----------------------*
C*      * Send success message *
C*      *----------------------*
C                   MOVEL     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                   MOVEL     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                   MOVEL     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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:            */
/* ADDLIBLE 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 "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];
 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 certiicate 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。

注: 法律に関する重要な情報については、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
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 subscripting 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*                **  Name of retained key
DNAME             S             64
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*
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                          9B 0
DRSNCOD                          9B 0
DEXTDTALN                        9B 0
DEXTDT                           4
DRARRYCT                         9B 0
DRARRY                          16
DKYNAM                          64
DHSHL                            9B 0
DHSH                            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
DHSHLEN                          9B 0
DHSH                            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 QMHSNDPM 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      PUBSECLEN = LENGTH
C                   EVAL      TKNINDEX = PUBSECLEN + 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      CRTSECLEN = 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 + CRTSECLEN)
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                   MOVEL     MSG(5)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     '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                   MOVEL     '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                   MOVEL     MSG(5)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     'CSNDPKH'     SAPI
C                   EXSR      SNDMSG
C                   ELSE
C*     *----------------------*
C*     *  Send success message *
C*     *----------------------*
C                   MOVEL     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: 公開鍵の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、295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:            */
```

```
/* ADDLIBLE 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 certiicate 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: 公開鍵の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、295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
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                     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                         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 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                          9B 0
DRSNCOD                          9B 0
DEXTDTALN                        9B 0
DEXTDT                           4
DRARRYCT                         9B 0
DRARRY                          16
DKYNAM                          64
DCRTLEN                          9B 0
DCRT                           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                               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             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                   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      PUBSECLEN = LENGTH
C                   EVAL      TKNINDEX = PUBSECLEN + 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      CRTSECLEN = LENGTH
C*
C*************************************************************
C* Register the public key
C*************************************************************
C*   *---------------------------------------*
C*   * Set the keywords in the rule array       *
C*   *---------------------------------------*
C                   MOVEL     '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                                            CRTSECLEN:
C                                            TOKENARRAY(TKNINDEX))
C*  *------------------------*
C*  * Check the return code  *
C*  *------------------------*
C     RETURNCODE    IFGT      0
C*     *-----------------------*
C*     *  Send failure message *
C*     *-----------------------*
C                   MOVEL     MSG(4)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   EXSR      SNDMSG
C                   ELSE
```

```
     C*     *----------------------*
     C*     *  Send success message *
     C*     *----------------------*
     C                   MOVEL     MSG(5)        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.
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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:            */
/* ADDLIBLE LIB(QCCA)                                                */
/* CRTCMOD MODULE(CERTKEY) SRCFILE(SAMPLE)                           */
/* CRTPGM  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 SAname[64];           /* Share administration or certifying    */
                            /* key name.                             */
 char SAnameASCII[64];      /* Share admin key name in ASCII         */
 long SAname_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(SAname, ' ', 64);  /* Make copy of Share Admin key name     */
memcpy(SAname,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 certiicate 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、295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
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 subscripting 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
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*                **  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
DHSH                            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
```

```
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
DHSHLEN                          9B 0
DHSH                            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      PUBSECLEN = LENGTH
C                   EVAL      TKNINDEX = PUBSECLEN + 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      CRTSECLEN = 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 + CRTSECLEN)
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                   MOVEL     SANAME        SANAMEASC
C                   CALL      'QDCXLATE'
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                   MOVEL     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                   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                   MOVEL     MSG(5)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     'CSNBOWH'     SAPI
C                   EXSR      SNDMSG
C                   RETURN
C                   ENDIF
C*   *---------------------------------------*
C*   * Set the keywords in the rule array       *
C*   *---------------------------------------*
C                   MOVEL     '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                   MOVEL     MSG(5)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     'CSNDDSG'     SAPI
C                   EXSR      SNDMSG
C                   RETURN
C                   ENDIF
C*
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 + PUBSECLEN + 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
```

例: マスター・キーの共用パーツを取得するための ILE C プログラム

マスター・キー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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*-------------------------------------------------------------------*/
/* GETSHARE                                                          */
/*                                                                   */
/* Sample program to obtain 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 4758 CCA Basic Services Reference and Guide           */
/*       (SC31-8609) publication.                                  */
/*                                                                 */
/* Parameters: Share number                                        */
/*             Name of share sender private key                    */
/*             Name of certifying key                              */
/*             Stream file containing receiver certificate         */
/*                                                                 */
/*                                                                 */
/* Example:                                                        */
/*   CALL PGM(GETSHARE) PARM(2 SENDR SAKEY RECVR.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 iSeries:          */
/* ADDLIBLE LIB(QCCA)                                              */
/* CRTCMOD MODULE(GETSHARE)  SRCFILE(SAMPLE)                       */
/* CRTPGM  PGM(GETSHARE)   MODULE(GETSHARE)                        */
/*         BNDDIR(QCCA/QC6BNDDIR)                                  */
/*                                                                 */
/* Note: Authority to the CSUAMKD service program                  */
/*       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 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 iSeries 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 certiicate 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,
         SAname,
         &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・キー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・キー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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
     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
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 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                 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                    SINDEX           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*       * 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      PUBSECLEN = LENGTH
C                   EVAL      TKNINDEX = PUBSECLEN + 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      CRTSECLEN = 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*   * 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                   MOVEL     '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                                            CRTSECLEN:
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                   MOVEL     MSG(4)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     'CSUAMKD'     SAPI
C                   EXSR      SNDMSG
C                   RETURN
C                   ENDIF
C*
C*************************************************************
C* Write share out to a file
C*************************************************************
C*    ** Build path name
C                   MOVEL     *ALLX'00'     PATH
C                   MOVEL     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・キーの共用パーツを導入するための ILE C プログラム

マスター・キーの共用パーツを導入するには、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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:           */
/* ADDLIBLE 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 "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 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 iSeries 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 certiicate 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・キーの共用パーツを導入するための ILE RPG プログラム

マスター・キーの共用パーツを導入するには、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、 295 ページの『第 6 章 コードについての特記事項』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 4758 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 iSeries:
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 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 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 QMHSNDPM 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                    SINDEX           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*       * 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      PUBSECLEN = LENGTH
C                   EVAL      TKNINDEX = PUBSECLEN + 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      CRTSECLEN = 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     SINDEX        SHAREIDX
C*    ** Build path name
C                   MOVEL     *ALLX'00'     PATH
C                   MOVEL     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                   MOVEL     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                   MOVEL     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                   MOVEL     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                   MOVEL     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                   MOVEL     MSG(5)        MSGTEXT
C                   EXSR      SNDMSG
C                   SETOFF                                         01
C                   ENDIF
```

```
C*
C                   CALLP     close         (FILED)
C  N01              SETON                                          LR
C*
C*************************************************************
C* Obtain a certificate
C*************************************************************
C*   *---------------------------------------*
C*   * Set share index number                *
C*   *---------------------------------------*
C                   Z-ADD     SINDEX        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                   MOVEL     '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                                            CRTSECLEN:
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                   MOVEL     MSG(6)        MSGTEXT
C                   MOVE      RETURNCODE    FAILRETC
C                   MOVE      REASONCODE    FAILRSNC
C                   MOVEL     '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                                          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 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: 保管キーをリストするための ILE C プログラム

保管キーをリストするには、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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*------------------------------------------------------------------*/
/* List the names of the RSA private keys retained within the       */
/* 4758.                                                            */
/*                                                                  */
/*                                                                  */
/*  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 4758 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 iSeries:            */
/* ADDLIBLE 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: 保管キーをリストするための ILE RPG プログラム

保管キーをリストするには、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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D*************************************************************
     D*
     D* List the names of the RSA private keys retained within the
     D* 4758.
     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 4758 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 iSeries:
```

```
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*-------------------------------------------------------------
DMSG              S             75    DIM(4) CTDATA PERRCD(1)
DMSGLENGTH        S              9B 0 INZ(75)
```

```
D                 DS
DMSGTEXT                  1     75
DNUMKEYS                  1      3
DNUMLABELS               25     26
DDSPLBL                   2     65
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* 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                   MOVEL     '*'           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    FAILRSNC
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                   MOVEL     LABELS(I)     DSPLBL
     C                   EXSR      SNDMSG
     C                   ENDFOR
     C*
     C                   ENDIF
     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
**
CSNDRKL failed with return/reason codes 9999/9999
There are no retained keys in the 4758
000 keys were found and 00 labels returned
[                                                                      ]
```

例: 保管キーを削除するための ILE C プログラム

保管キーを削除するには、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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/* Delete a retained key from the 4758                                */
/*                                                                    */
/*                                                                    */
/*  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 4758 CCA Basic Services Reference and Guide             */
/*       (SC31-8609) publication.                                    */
/*                                                                   */
/* Parameters:                                                       */
/*   none.                                                           */
/*                                                                   */
/* Example:                                                          */
/*   CALL PGM(DLTRTNKEY) (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 iSeries:            */
/* ADDLIBLE LIB(QCCA)                                                */
/* CRTCMOD MODULE(DLTRTNKEY)  SRCFILE(SAMPLE)                        */
/* CRTPGM  PGM(DLTRTNKEY)  MODULE(DLTRTNKEY)                         */
/*         BNDSRVPGM(QCCA/CSNDRKD)                                   */
/*                                                                   */
/* Note: Authority to the CSNDRKD service program in the             */
/*       QCCA library is assumed.                                    */
/*                                                                   */
/*                                                                   */
/*-------------------------------------------------------------------*/
#include <string.h>
#include <stdio.h>
#include "csucincl.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: 保管キーを削除するための ILE RPG プログラム

保管キーを削除するには、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、 295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
     D************************************************************
     D* DLTRTNKEY
     D*
     D* Sample program to delete a retained key from the 4758
     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 4758 CCA Basic Services Reference and Guide
D*       (SC31-8609) publication.
D*
D* Parameters:
D*   Retained key label name
D*     (64 chacters - 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 iSeries:
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
```

4758 暗号化コプロセッサーのトラブルシューティング

4758 コプロセッサー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応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、4758 コプロセッサーは、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の場合があります。
- **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、4758 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)、iSeries のライセンス内部コード、または 4758 コプロセッサー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。

共通のエラー

次のような共通のエラーにはよく注意する必要があります。

- **4758 コプロセッサーと CCA CSP を導入したか** 2620 暗号プロセッサーと 2628 暗号プロセッサー - 商用は、IBM CCA Services for iSeries で動作する、完

全に別個のソリューションです。 4758 コプロセッサーと CCA CSP は、2620、2628、または IBM CCA サービスでは動作しません。

- **装置をオンに変更したか** 装置をオンに変更するまでは、4758 コプロセッサーには、いかなる要求も送ることができません。
- **暗号装置記述の資源は、4758 暗号化コプロセッサーか。それとも 2620 または 2628 か。** 4758 暗号化コプロセッサーでなければなりません。
- **4758 コプロセッサー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 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。
- **正しい装置を選択しているか** 省略時の装置 (たとえば、CRP01 という名前の装置) と追加装置がある場合には、4758 コプロセッサーは、ユーザーが Cryptographic_Resource_Allocate を使用しないかぎり、省略時の装置が選択されます。
- **4758 コプロセッサーは鍵ストア・ファイルを検出しているか** Key_Store_Designate SAPI を明示的に使用した場合には、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す。装置記述でファイルに名前を付けていない場合には、4758 コプロセッサーはすべてのファイルを検出しません。
- **マスター・キーをロードして設定しているか** 4758 コプロセッサーは、マスター・キーをロードしない限り、4758 コプロセッサー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。
- **古いマスター・キー・レジスターにキーが含まれているか** 4758 コプロセッサーは、古いマスター・キー・レジスターに値がないかぎり、現行のマスター・キーではキーを再暗号化することができません。
- **省略時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。
- **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** 4758 コプロセッサーがハードウェア・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、4758 コプロセッサーを再初期化する必要があります。再初期化は、Cryptographic_Facility_Control API またはシステム・サービス・ツールにあるハードウェア・サービス管理プログラムを使用します。 Cryptographic_Facilty_Control API を使用するには、4758 コプロセッサー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す。このような役割が存在しない場合、ハードウェア・サービス管理プログラムを使用します。
- **機能制御ベクトルがロードされているか** 4758 コプロセッサーは、機能制御ベクトルをロードするまでは、構成以外の暗号操作を実行することはできません。
- **Cryptographic Access Provider 製品の 1 つが導入されているか** IBM は、機能制御ベクトルをこれらのプロダクトと共に出荷します。

- **マスター・キーをロードしている場合には、新規のマスター・キー・レジスターを削除してから開始したか** 4758 コプロセッサーが新規のマスター・キー・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・キーの最初の部分をロードすることはできません。
- **実行する権限を DEFAULT の役割から削除する前に、4758 でクロックを設定したか** 設定していない場合、Cryptographic_Facility_Control API またはシステム・サービス・ツールにあるハードウェア・サービス管理を使用して 4758 コプロセッサーを初期化します。Cryptographic_Facilty_Control API を使用するには、4758 コプロセッサー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す。このような役割が存在しない場合、ハードウェア・サービス管理プログラムを使用します。
- **公開鍵 - 秘密鍵ペアを生成する前に EID を設定したか** RSA キーを生成する前に EID を設定する必要があります。
- **ヌル・キー・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 サポートは最初のバイトをキー・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ラベル形式としてレポートするか、またはキー・レコードを検出できたとレポートします。
- **PKA 鍵ストア・ファイルおよび保管されている PKA キーにあるラベルに対して同じ名前を使用したか** 同じ名前を使用した場合には、4758 コプロセッサーは、鍵ストア・ファイルを最初に検索するため、保管されているキーを検出することはありません。
- **スケルトン PKA キー・トークンのいずれかのフィールドに EBCDIC データがあるか。** 4758 コプロセッサーは、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。

より詳細なトラブルシューティング情報については、『4758 暗号化コプロセッサーの再初期化』と 286 ページの『ハードウェア・サービス管理プログラムの使用』を参照してください。

4758 暗号化コプロセッサーの再初期化

4758 コプロセッサー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。たとえば、マスター・キー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。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。

- 280 ページの『例: 4758 コプロセッサーを再初期化するための ILE C プログラム』
- 283 ページの『例: 4758 コプロセッサーを再初期化するための ILE RPG プログラム』

注: 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それを使用することをお勧めします。

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、286 ページの『ハードウェア・サービス管理プログラムの使用』で説明されているシステム・サービス・ツールのハードウェア・サービス管理プログラム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。

4758 コプロセッサー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 4758 コプロセッサーにロードすると、4758 コプロセッサーに保管されているマスター・キー、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 4758 コプロセッサー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・キー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・キーをランダムに生成した場合には、そのキーを 2 番目の 4758 コプロセッサーに複製する必要があります。複製しない場合、4758 コプロセッサーを再初期化したときに暗号化されたキーがすべて失われます。

例: 4758 コプロセッサーを再初期化するための ILE C プログラム

4758 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
/*--------------------------------------------------------------------*/
/* Clear the 4758 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 4758 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 iSeries:           */
/* ADDLIBLE 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_Facilitiess_Control (CSUACFC).                     */
/*                                                                  */
/*------------------------------------------------------------------*/


#include "csucincl.h"     /* header file for CCA Cryptographic      */
                          /* Service Provider for iSeries           */
#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("%ld/%ld¥n¥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("4758 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 4758 ");

    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: 4758 コプロセッサーを再初期化するための ILE RPG プログラム

4758 コプロセッサー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、295 ページの『第 6 章 コードについての特記事項』を参照してください。

```
D*************************************************************
D* REINIT
D*
D* Clear the 4758 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 4758 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 iSeries:
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_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             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_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                   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                 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*-------------------------------------------------------------*
C* Set the keyword in the rule array                          *
C*-------------------------------------------------------------*
C                   MOVEL     '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 Facilty 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                   RETURN
C                   ENDIF
C*
C*         *--------------------------------------*
C*         * Send success message for the 1st step *
C*         *--------------------------------------*
C                   MOVEL     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 Facilty 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(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
**
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・サービス管理プログラムは、論理的およびパッケージの両方の観点から、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) プロセッサーのデバッグを支援するものです。また、4758 コプロセッサー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)。

4758 コプロセッサーが再初期化されると、4758 コプロセッサー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。コプロセッサー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・サービス管理プログラム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・キー、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。

4758 コプロセッサー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。たとえば、コプロセッサーの構成が正しくない場合には、コプロセッサーの便利な機能をまったく実行できなかったり、4758 コプロセッサー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・サービス管理プログラムは、システム・サービス・ツールに含まれています。システム・サービス・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。「サービス・ツールの開始 (STRSST) のサインオン」画面が表示されます。

```
                    サービス・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。

  サービス・ツール・ユーザー     ......
  サービス・ツール・パスワード ......









F3= 終了     F9= パスワード変更     F12= 取り消し
```

サービス・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・サービス・ツール」画面が表示されます。

```
                        システム・サービス・ツール (SST)


次の中から 1 つを選んでください。

    1. サービス・ツールの開始
    2. 活動サービス・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
```

サービス・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。「サービス・ツールの開始」画面が表示されます。

```
                          サービス・ツールの開始

Warning: Incorrect use of this service tool can cause damage
to data in this system. Contact your service representative
for assistance.

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
```

ハードウェア・サービス管理プログラムを開始するには、**7** を選択してください。ハードウェア・サービス管理プログラムの画面に、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，技術員専用に提供されています。

   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
```

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. プロセッサー資源
   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
```

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，実行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

                                                                リソース
Opt Description                Type-Model  Status              Name
 _    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
```

4758 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 4758 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，実行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

                                                                リソース
Opt Description                      Type-Model  Status          Name
  _   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
```

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
```

フラッシュ・メモリーを再初期化する (4758 コプロセッサー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キーに関して必要な予防措置を取り、マスター・キー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

危険：

 Performing this initialization of the flash memory on the cryptography
 device will cause ALL key information stored on the device to be
 DESTROYED. This will cause all data encrypted using this device to be
 rendered unusable.


警告：

 Performing this initialization of the flash memory on the cryptography
 device will take an estimated 10 minutes.




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・サービス・ツールを終了します。

第 5 章 暗号化ハードウェアの関連情報

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 レッドブック™

- 「IBM @server iSeries Wired Network Security: OS/400 V5R1 DCM and Cryptographic Enhancements」

IBM の資料

- 「IBM 暗号化ハードウェア」。このサイトには、iSeries サーバー用の、4758 暗号化コプロセッサーのハードウェア・ソリューションについての情報が記載されています。
- 「CCA Basic Services Manual」。これは、IBM 4758 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 PCI Cryptographic Coprocessor documentation library」。ここには、4758 暗号化コプロセッサーの一般情報、サポート情報、およびプログラミング情報が記載された、ダウンロード可能な PDF 資料が含まれています。
- 「IBM developerWorks」。このサイトには、暗号の概念、暗号方式、および暗号化についての解説が記載されています。

第 6 章 コードについての特記事項

本書には、プログラムの例が含まれています。

IBM は、お客様に、このプログラムをサンプルとして使用することができる非独占的な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。

このサンプル・コードは、例として示す目的でのみ、IBM により提供されます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。

ここに含まれるすべてのプログラムは、現存するままの状態で提供され、いかなる保証条件も適用されません。不侵害、商品性、特定目的適合性に関する黙示の保証の適用も一切ありません。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12