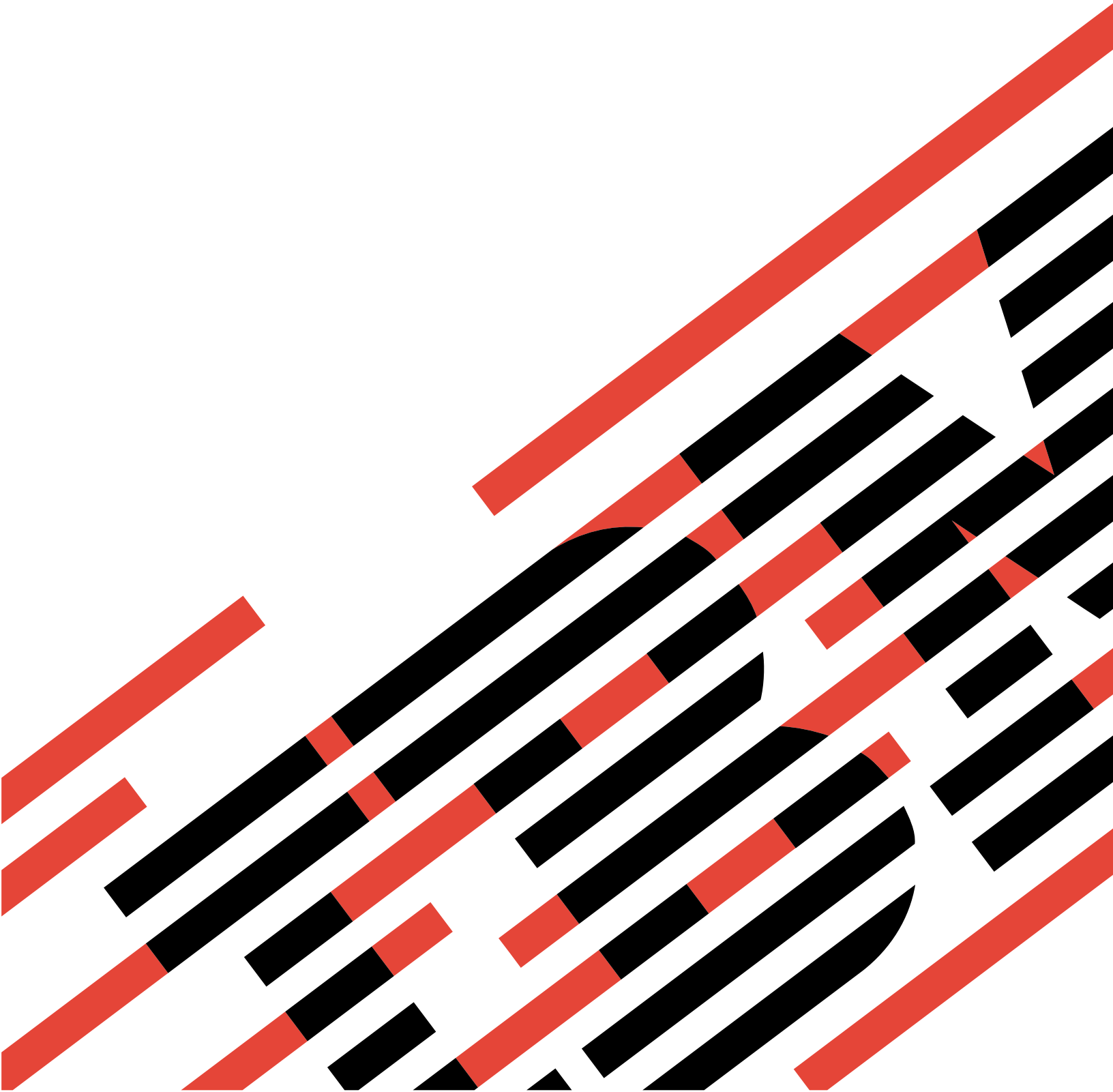
IBM

eserver

iSeries

加密硬體



IBM

@server

iSeries

加密硬體

© Copyright International Business Machines Corporation 1998, 2003. All rights reserved.

目錄

第 1 篇 4758 加密輔助處理器 1

第 1 章 列印此主題 3

第 2 章 V5R2 的新增功能 5

第 3 章 概念 7

第 4 章 iSeries 的 4758 加密輔助處理器 11

4758 加密輔助處理器的特性 12
4758 加密輔助處理器的加密硬體實務 13
加密硬體實務：使用加密硬體以保護私密金鑰 . . 13
加密硬體實務：撰寫 OS/400 應用程式來使用 4758 加密輔助處理器 14
4758 加密輔助處理器的計畫 16
4758 加密輔助處理器的基本要求 16
對 4758 加密輔助處理器的安全存取 17
配置 4758 加密輔助處理器 20
建立裝置說明 21
爲金鑰儲存檔之檔案命名 21
建立及定義角色和設定檔 22
設定環境 ID 與時鐘 59
載入函數控制向量 70
載入及設定主要金鑰 82
配置 4758 加密輔助處理器以與 DCM 及 SSL 一起使用 93
配置 4758 加密輔助處理器以與 OS/400 應用程式一起使用 94
移轉至 4758 加密輔助處理器 94
由其它 iSeries 加密產品移轉至「4758 加密輔助處理器」 94
移轉金鑰儲存檔案 120
管理 4758 加密輔助處理器 120
登入或登出 4758 加密輔助處理器. 120
查詢狀態或要求資訊 131
起始設定金鑰儲存檔案 136
建立 DES 與 PKA 金鑰. 141
加密或解密檔案. 148
使用 PIN 154
產生並驗證數位簽章 167
管理多重 4758 加密輔助處理器 176
複製主要金鑰 186
4758 加密輔助處理器疑難排解 263
重新起始設定 4758 加密輔助處理器 264
使用硬體服務管理程式 271

第 5 章 加密硬體的相關資訊 279

第 6 章 程式碼不保事項聲明 281

© Copyright IBM Corp. 1998, 2003

第 1 篇 4758 加密輔助處理器

© Copyright IBM Corp. 1998, 2003

第 1 章 列印此主題

您可檢視或下載這些主題的 PDF 版本：

- 加密硬體 (大約 756 KB 或 292 頁) 包含所有關於 V5R2 上 iSeries™ 伺服器所支援之 IBM® 加密硬體的資訊。
- 4758 加密輔助處理器 (大約 753 KB 或 296 頁) 包含關於 V5R2 上 iSeries 伺服器所支援之「4758 加密輔助處理器」硬體的資訊。

儲存 PDF 檔案

若要在您的工作站上儲存 PDF 以供檢視或列印：

1. 在您的瀏覽器中以滑鼠右鍵按一下 PDF。
2. 按一下**另存目標**。
3. 導覽到您想要儲存 PDF 之處。
4. 按一下**儲存**。

下載 Adobe Acrobat Reader

如果您需要 Adobe Acrobat Reader 來檢視或列印這些 PDF，可從 Adobe 網站 (www.adobe.com/products/acrobat/readstep.html) 下載複本。

© Copyright IBM Corp. 1998, 2003

第 2 章 V5R2 的新增功能

若您正在尋找關於最新加密硬體之最新資訊，及 iSeries 伺服器之現有加密硬體選項的新增特性，那麼您是來對地方了。

新的加密硬體：IBM 2058 電子商業加密加速器

除了「4758 加密輔助處理器」外，也可用 IBM 2058 電子商業加密加速器 (「硬體特性」碼 4805，此後稱爲「2058 加密加速器」)。透過重新遞送私密金鑰的處理程序 (使它不由系統處理器處理) 的設計，來增進 iSeries 的效能，此硬體選項爲處理高容量 SSL (Secure Sockets Layer) 交易的 iSeries 施行之極好選擇。雖然「2058 加密加速器」是增強 iSeries 伺服器 SSL 效能的極好選擇，並且容易安裝及初始化，但是它並未提供「4758 輔助處理器」所提供的大量配置選項。

附加功能：「4758 加密輔助處理器」

「4758 加密輔助處理器」向客戶提供了下列新功能：

- 金融個人識別碼 (PIN) 處理程序：每筆交易唯一金鑰 (UKPT)
- 共同密碼架構 (CCA) 2.4

新的加密硬體實務

爲了向您提供關於如何與 iSeries 伺服器一起使用加密硬體的一些意見，我們已經將下列實務新增至 「iSeries 資訊中心」：

- 加密硬體實務：以加密硬體保護私密金鑰
- 加密硬體實務：撰寫 OS/400® 應用程式以使用「4758 加密輔助處理器」

若要尋找此版次中有關新增功能或變更的其它資訊，請參閱使用者備忘錄

。

如何查看新增功能或變更

爲了協助您查看已作過技術變更的地方，此資訊使用：

- 影像標示新增功能或變更資訊開始的地方。
- 影像標示新增功能或變更資訊結束的地方。

© Copyright IBM Corp. 1998, 2003

第 3 章 概念

加密

加密之於資料保全，不僅是一門技術，也是一種藝術。基本的加密服務必須能夠確保訊息的隱密性、維護訊息的完整性，並鑑別通信各方的身份，以及保證通信方無法否認其曾經傳送過訊息。

您可以利用加密來儲存資訊或與其他方進行通信，並保證未參與的各方無從了解儲存的資訊或通信的內容。加密會將可以理解的文字轉換成無法理解的資料 (密碼文字)。而解密則會將這些無法理解的資料復置成可以理解的文字。兩種處理都和數學公式、演算法及密鑰資料 (金鑰) 有密切的關係。

加密演算法

加碼演算法有下列兩種類型：

1. 使用密鑰或**對稱金鑰演算法**的通信雙方會共用同一只金鑰。加密和解密都必須使用該份金鑰。「資料加密標準 (DES)」與「三重 DES 演算法」皆屬密碼鎖演算法的例子。
2. 公開金鑰或**不對稱金鑰演算法**則是使用一對金鑰。其中的私密金鑰爲單方專用，不會與他人共享。而公開金鑰則不然，會公開與他人共用。資料在使用其中一只金鑰加密之後，便須使用另一只金鑰，才能夠予以解密及回復。上述兩只金鑰雖然都是以數理爲基礎，但實際上絕無可能從公開金鑰推得私密金鑰。RSA 演算法爲公開金鑰演算法一種。

兩種演算法類型都會透過金鑰來決定資料的變更方式。不同的加密處理會依其所要達成的目的而選用適合的演算法。您可以根據您的目的選擇所需的加密處理；例如您可能需要產生訊息鑑別碼 (MAC)，以確保資料的完整性。使用者針對「4758 加密輔助處理器」所撰寫的應用程式，會使用相對應的安全性應用程式設計介面 (SAPI) 來呼叫加密處理。金鑰和加密處理會一起轉換資料。具有 SAPI 授權的使用者將可以存取該項加密處理。由此可知，金鑰控制了資料的存取權限。您必須保護金鑰，才能夠保護資料。您若能夠妥善保管金鑰值而不外洩，便可以保障每一次使用該金鑰演算加密之資料的安全性。

加密

對於欄位層次的加密，使用者應用程式會明確地要求加密服務。使用者應用程式全權控制金鑰的產生、選取及分送。使用者應用程式同時還控制了加密的資料項目，以及應保留爲純文字的資料項目。對於階段作業層次的加密，則系統會要求加密服務，而不是應用程式。您的應用程式可能知道，也可能不知道加密的進行。鏈結層次的加密通常會利用加密專用的硬體，在通信協定堆疊的最底層執行。「4758 輔助處理器」支援欄位層次的加密及 Secure Sockets Layer (SSL) 階段作業建立的加密，但不支援 VPN 或 SNA 階段作業層次的加密。「2058 加密加速器」只支援 SSL 階段作業建立的加密。

資料完整性

© Copyright IBM Corp. 1998, 2003

要確定資料的可靠與否，您必須了解資料是否出自於授權的來源，且未經變更。亦即資料的正確性及資料的完整性。「4758 輔助處理器」藉建立「訊息鑑別碼 (MAC)」、訊息摘要或數位簽章來確定資料的正確性與整合性。

訊息鑑別碼 (MAC)

MAC 處理是一種保持資料完整的技術，可以讓您定義重要的資料元素。例如，您可以定義資金轉送訊息中的金額部份。重要資料元素、密碼演算法與 MAC 金鑰會產生 MAC。之後 MAC 即成爲訊息的一部份，且與訊息一起遞送。MAC 處理會使用 DES 或「三重 DES 演算法」金鑰。

訊息接收者會使用與傳送者相同的 MAC 金鑰、演算法及程序重新產生 MAC。若接收者的 MAC 與訊息一起傳送的 MAC 相符，便可以原封不動地收下該 MAC。

MAC 處理可協助鑑別收到的訊息；但由於傳輸的資料爲純文字格式，因此無法防堵未經授權的讀取進行。藉使用 MAC 處理進而加密整個訊息，將可以更有效地保障資料的私密性與完整性。

訊息摘要

您可以對資料執行訊息摘要處理，以產生摘要值，作爲加密過程中所產生的總和檢查。資料一經修改，所產生的摘要亦會有所不同。您可以保留訊息摘要的複本加以比較。訊息摘要若是相同，便表示資料未經修改。

數位簽章

數位簽章也可用於驗證正確性及完整性。此處理一共有兩個步驟：

1. 首先產生資料的摘要，然後再使用 RSA 私密金鑰加密摘要。其結果便是數位簽章。您可以使用公開金鑰解密簽章，回復其原始摘要，對簽章進行驗證。
2. 產生另一份資料摘要，與原始摘要進行比較。若兩者相同，便表示通過驗證，可以確信該資料未經任何改變。

「4758 加密輔助處理器」相關的金鑰類型

「4758 輔助處理器」所使用的金鑰類型繁多。但並非所有的 DES 或「三重 DES 演算法」金鑰都適用於全部的對稱金鑰作業。同樣地，也並非所有的公開金鑰演算法 (PKA) 金鑰，都適用於全部的不對稱金鑰作業。以下是「4758 輔助處理器」所使用的金鑰類型清單：

主要金鑰

此爲未加密金鑰，表示其未經其它金鑰的加密。「4758 輔助處理器」利用主要金鑰加密所有的作業金鑰。「4758 輔助處理器」會將主要金鑰存在損害回應模組中。您無法從「4758 輔助處理器」中擷取之。「4758 輔助處理器」會以損毀主要金鑰及其原廠證明的方式回應損害的企圖。4758-023 有兩只主要金鑰：一只用於加密 DES 金鑰，另一只則用於加密 PKA 金鑰。

雙倍長度的加密用的密鎖鑰

「4758 輔助處理器」會使用這種類型的「三重 DES 演算法」金鑰，加密或解密其它的 DES 或「三重 DES 演算法」金鑰。加密用金鑰一般會用於系統之間的金鑰傳輸。但也可以在離線時儲存金鑰以爲備份。若使用加密用金鑰傳輸金鑰，則兩個系統便須共用加密用金鑰本身的清除值。匯出器的加密用金鑰會於匯出作業時使用，並於其間將使用

主要金鑰加密的金鑰解密，接著再以加密用的金鑰予以加密。匯入器的加密用金鑰會在匯入作業時使用，並於其間將使用加密用金鑰加密的金鑰解密，接著再以主要金鑰予以加密。

雙倍長度的 PIN 金鑰

「4758 輔助處理器」會透過此種類型的金鑰產生、驗證、加密及解密金融作業中所使的 PIN。這些屬於「三重 DES 演算法」金鑰。

MAC 金鑰

「4758 輔助處理器」會使用此種類型的金鑰產生「訊息鑑別碼 (MAC)」。這些可以是 DES 或「三重 DES 演算法」金鑰。

密碼金鑰

「4758 輔助處理器」會使用此種類型的金鑰加密或解密資料。這些可以是 DES 或「三重 DES 演算法」金鑰。

單倍長度相容性金鑰

「4758 輔助處理器」會使用此種類型的金鑰加密或解密資料，並產生 MAC。這些是 DES 金鑰，多在加密資料或 MAC 與未施行「共同密碼架構」的系統進行交換時使用。

私密金鑰

「4758 輔助處理器」會利用私密金鑰來產生數位簽章，以及解密由公開金鑰所加密的 DES 或「三重 DES 演算法」金鑰。

公開金鑰

「4758 輔助處理器」會使用公開金鑰來驗證數位簽章、加密 DES 或「三重 DES 演算法」金鑰，以及解密由私密金鑰所加密的資料。

金鑰格式

「4758 輔助處理器」會使用下列四種金鑰格式之一。金鑰格式及金鑰類型決定了加密處理運用該金鑰的方式。這四種格式包括：

清除格式

金鑰的清除值未以任何加密的方法保護。「4758 輔助處理器」不使用未加密金鑰。未加密金鑰必須先匯入安全的模組中，並使用主要金鑰加密，然後後儲存於安全模組之外。

作業格式

以主要金鑰加密的金鑰屬作業格式。「4758 輔助處理器」的加密作業可以直接使用這些金鑰。作業金鑰又稱爲內部金鑰。所有儲存在伺服器金鑰儲存檔中的金鑰都屬於作業金鑰。但您無需將所有的作業金鑰都儲存在金鑰儲存檔中。

匯出格式

以匯出器加密用金鑰加密的金鑰作爲匯出作業結果者皆屬匯出格式。這類金鑰又稱爲外部金鑰。若匯入器加密用金鑰的清除值與匯出器加密用金鑰的清除值相同，則此種匯出格式的金鑰也可以歸類成匯入格式。您可以選擇任意方式將金鑰儲存成匯入格式，但卻不可以將其存入金鑰儲存檔中。

匯入格式

以匯入器加密用金鑰加密的金鑰皆屬匯入格式。只有匯入格式的金鑰才可以用爲匯入作業的來源。這類金鑰又稱爲外部金鑰。若匯出器加密用金鑰的清除值與匯入器加密用金鑰的清除值相同，則此種匯入格

式的金鑰也可以歸類成匯出格式。您可以選擇任意方式將金鑰儲存成匯出格式，但卻不可以將其存入金鑰儲存檔中。

函數控制向量

IBM 提供的數位簽章值稱爲「函數控制向量」。該值將促使「4758 輔助處理器」中的加密應用程式，產生與適用之匯入規則與匯出規則一致的加密服務層次。「函數控制向量」會隨附在您系統上所安裝的 IBM Cryptographic Access Provider (5722-ACx) 產品中。該檔案的路徑名稱爲 /QIBM/ProdData/CAP/FCV.CRT。函數控制向量提供「4758 輔助處理器」產生金鑰所需的金鑰長度資訊。

控制向量

控制向量與函數控制向量不同，是已知的值，與控制下列項目的金鑰相關聯：

- 金鑰類型
- 其它可由此金鑰加密的金鑰
- 「4758 輔助處理器」是否可以匯出此金鑰
- 此金鑰所允許的其它用途

控制向量會以加密的方式鏈結到某只金鑰，若未同時變更此金鑰值，便無法變更此控制向量。

金鑰儲存檔

爲 OS/400 資料庫檔案，用於儲存以「4758 輔助處理器」之主要金鑰加密的金鑰。

金鑰記號

一種資料結構，內含加密金鑰、控制向量及其它金鑰相關資訊。大部份作用於金鑰上或使用金鑰的 CCA API 動詞都會將金鑰記號用爲參數。

第 4 章 iSeries 的 4758 加密輔助處理器

使用 iSeries 伺服器，可以按下列方式使用「4758 輔助處理器」：

- 您可以與 DCM 一起使用「4758-023 輔助處理器」來產生並儲存與 SSL 數位憑證相關的私密金鑰。此外，於 SSL 階段作業建立期間，「4758-023 輔助處理器」透過處理 SSL 私密金鑰處理程序來提供效能輔助加強功能和能力。
- 爲了支援平衡資料流量和效能調整大小， iSeries 伺服器容許使用多重 (最多達到八個) 具有 SSL 的「4758-023 輔助處理器」。當使用多重「輔助處理器」時，爲使用硬體來產生並儲存與數位憑證相關的私密金鑰， DCM 配置提供下列選項。
  1. 私密金鑰在硬體上產生並儲存 (即保留) 於硬體上。

     使用此選項，私密金鑰從未脫離「輔助處理器」，因而無法與另一個「輔助處理器」一起使用或共用。這表示您和您的應用程式必須管理多重私密金鑰和憑證。
  2. 私密金鑰產生於硬體上，而儲存於軟體上 (即儲存在金鑰儲存檔案中)。

     此選項容許在多重「輔助處理器」中共用單一私密金鑰。基本要求是每一個「輔助處理器」必須共用相同的主要金鑰 --您可以使用第 186 頁的『複製主要金鑰』來設定「輔助處理器」，以使之有相同的主要金鑰。此私密金鑰在某一個「輔助處理器」中產生，然後儲存在金鑰儲存檔案中，並在該「輔助處理器」的主要金鑰下加密。任何有相同主要金鑰的「輔助處理器」都可以使用該私密金鑰。
- 您可以使用「4758-023 輔助處理器」來執行 OS/400 應用程式。若要這樣做，您或應用程式提供者必須使用 CCA CSP API 來存取「4758 輔助處理器」中的加密服務，以撰寫應用程式。這個應用程式的範例爲關於金融個人識別碼 (PIN) 處理交易、銀行至票據交換所交易與基本的 SET™ 大量處理。透過 CCA CSP 最多可以使用八個「輔助處理器」。應用程式必須藉由使用 Cryptographic_Resource_Allocate (CSUACRA) 和 Cryptographic_Resource_Deallocate (CSUACRD) CCA API 來控制對個別「輔助處理器」的存取。

「4758 輔助處理器」的相關資訊，請參照下列頁：

第 12 頁的『4758 加密輔助處理器的特性』

「4758 輔助處理器」包含硬體引擎，該硬體引擎執行 SSL 及 OS/400 應用程式期間使用的加密作業。

將「共同密碼架構密碼服務提供程式 (CCA CSP)」套裝爲 OS/400 選項 35。它提供安全性應用程式設計介面 (SAPI) 以撰寫容許存取「4758 輔助處理器」密碼服務的應用程式。

特性頁將會更詳細地說明「4758 輔助處理器」和 CCA CSP 必須提供的服務。

第 16 頁的『4758 加密輔助處理器的基本要求』

在可以安裝和使用「4758 輔助處理器」之前，您的伺服器必須符合部份基本要求。使用基本要求頁來確定是否備妥在伺服器上安裝和使用「4758 輔助處理器」。

第 7 頁的第 3 章, 『概念』

© Copyright IBM Corp. 1998, 2003

取決於您對加密的熟悉程度，您可能需要更多關於術語或概念的資訊。此頁介紹部份基本加密概念。

請參閱相關資訊，以獲取由 IBM 建議之加密資訊的附加來源檔。

4758 加密輔助處理器的特性

「4758 PCI 加密輔助處理器」提供加密處理能力及加密金鑰的安全儲存體。受支援的加密功能包括保持資料機密的加密/解密、確定資料沒有被變更的訊息摘要及訊息鑑別碼 (MAC)、數位簽章產生/驗證和金融個人識別碼 (PIN) 及 SET 處理。您可以使用具有 OS/400 SSL，或具有由您或應用程式提供者撰寫之 OS/400 應用程式的「輔助處理器」。

當與 SSL 一起使用時，「輔助處理器」可以用來在 FIPS 140-1 已認證的硬體模組中建立和儲存私密金鑰，也可以使用「輔助處理器」在軟體上建立、加密並儲存私密金鑰 (在主要金鑰下加密)，以使多重「輔助處理器」卡片可以使用此私密金鑰。主要金鑰總儲存於 FIPS 140-1 已鑑定的硬體模組中。此外，在 SSL 階段作業的建立期間，「輔助處理器」將從 iSeries 伺服器的主要 CPU 卸載加強計算性的加密處理程序。

對於 OS/400 應用程式，「共同密碼架構密碼服務提供程式 (CCA CSP) API」用來存取「輔助處理器」的加密和金鑰管理功能。

「4758-023 PCI 加密輔助處理器」支援 DES、三重 DES 演算法、RSA、MD5、SHA-1、 RIPEMD-160 和財務個人識別碼 (PIN) 服務。另外，它支援 SET (「安全電子交易」) 區塊加密服務。「4758 輔助處理器」的主要好處是它提供儲存加密金鑰的功能。它以損害回應和電池備份模組 (亦稱爲安全模組) 來實現這個功能。「4758-023 PCI 加密輔助處理器」「聯邦資訊處理標準 (FIPS) PUB 1400-1，第三級」的基本要求。 「4758 輔助處理器」的另一個好處是，在 SSL 階段作業建立期間，它可以從計算性加強的加密處理程序中卸載 iSeries 伺服器的主要 CPU。

「4758 輔助處理器」提供角色型的存取控制機能，這可讓您對「輔助處理器」支援的個別加密作業的存取權限進行啓用和控制。

CCA CSP 特性

您可以使用含有 CCA CSP 的「4758 輔助處理器」來爲應用程式提供高層次的加密安全性。客戶或協力廠商應用程式透過一套應用程式設計介面 (API) 來存取這些服務。您可以在 IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide 中找到這些 API 的說明 API 與「OS/400 選項 35 - 共同密碼架構密碼服務提供程式 (CCA CSP)」一起被提供。

「4758 輔助處理器」將使用 CCA 主要金鑰來加密金鑰，以便您可以在「4758 輔助處理器」之外儲存那些金鑰。您將儲存那些金鑰於金鑰儲存檔案中，該金鑰儲存檔案是資料庫檔案。

IBM 的 CCA 在主要 IBM 計算平台上啓用一致的加密方法。 OS/400 CCA CSP 支援以 ILE C、RPG 和 Cobol 撰寫的應用軟體。應用軟體可以呼叫 CCA 服務，以執行大範圍的加密功能，包括了「資料加密標準 (DES)」及 RSA 演算法。

OS/400 CCA CSP 提供相當於「CCA 支援程式」的 API 支援，該支援程式可用於 NT、AIX® 及 OS/2®。CCA CSP 開拓「4758 輔助處理器」的能力，並可讓您完成下列步驟：

- 建立角色型的存取控制來定義提供給使用者的存取層次。
- 產生隨機數字。
- 安全地複製主要金鑰。
- 支援財務 PIN 處理程式。
- 建立和驗證數位簽章。
- 加密和解密資料。
- 保護金鑰。
- 安全地匯入和匯出已加密的 DES 和三重 DES 演算法金鑰。
- 建立「訊息鑑別碼 (MAC)」。

4758 加密輔助處理器的加密硬體實務

若要向您提供有關如何搭配使用 iSeries 伺服器與加密硬體的部份意見，我們已新增下列用法實務：

- 加密硬體實務：以加密硬體保護私密金鑰

  對於需要增加與安全 SSL 商業交易相關的 iSeries 伺服器數位憑證私密金鑰之安全性的公司，此實務可能非常有用。
- 加密硬體實務：撰寫 OS/400 應用程式以使用「4758 加密輔助處理器」

  此實務可以 OS/400 程式設計師思考如何撰寫程式呼叫「4758 加密輔助處理器」以驗證使用者資料，例如在自動提款機 (ATM) 上輸入的財務個人識別碼 (PIN)。

加密硬體實務：使用加密硬體以保護私密金鑰

狀況

某公司有專用於處理企業對企業的電子商業 (B2B) 交易的 iSeries 伺服器。此公司管理人員已通知 iSeries 伺服器專員 Sam，有關其 B2B 客戶之一項安全性基本要求。基本要求爲增加 iSeries 伺服器數位憑證私密金鑰的安全性，其與 Sam 的公司所執行之安全 SSL 商業交易相關。Sam 聽說可以使用 iSeries 伺服器上的加密硬體選項，加密與儲存和篡改回應硬體上的 SSL 交易相關的私密金鑰：「IBM 4758-023 加密輔助處理器」PCI 卡 (硬體特性碼 4801 或 4802，此後亦稱爲「4758 加密輔助處理器」)。

Sam 研究「4758 加密輔助處理器」，得知它可以與 OS/400「數位憑證管理程式 (DCM)」一起使用，以提供安全 SSL 私密金鑰儲存體，也可以藉由從 iSeries 伺服器卸載已在建立 SSL 階段作業期間完成的加密作業，以增加 iSeries 伺服器效能。

註: 若要支援平衡資料流量和效能調整大小，Sam 可於 iSeries 伺服器上使用含有 SSL 的多重 (最多達到八個)「4758 加密輔助處理器」。請參閱 iSeries 上的實施多重加密硬體卡，以獲得更多資訊。

Sam 認爲「4758 加密輔助處理器」符合增加公司 iSeries 伺服器的安全性基本要求。

明細

1. 公司的 iSeries 伺服器已安裝和配置「4758 加密輔助處理器」，以儲存及保護私密金鑰。
2. 私密金鑰爲「4758 加密輔助處理器」所產生。
3. 然後在「4758 加密輔助處理器」上儲存私密金鑰。
4. 「4758 加密輔助處理器」會防止實體及電子駭客入侵的嘗試。

先決條件與假設

1. iSeries 伺服器已適當地安裝和配置的「4758 加密輔助處理器」 (請參閱「4758 加密輔助處理器」的規劃及配置「4758 加密輔助處理器」)。「4758 加密輔助處理器」的規劃包括在 iSeries 伺服器上執行 SSL。

   註: 若要爲應用程式 SSL 交握式處理程序和保護私密金鑰使用多重「4758 加密輔助處理器」卡， Sam 將需要確定他的應用程式可以管理多重私密金鑰和憑證。
2. Sam 的公司已安裝和配置「數位憑證管理程式 (DCM)」，並使用它來管理 SSL 通信階段作業的公用網際網路憑證。
3. Sam 的公司從公用「認證權限 (CA)」獲得憑證。
4. 使用 DCM 之前已轉接「4758 加密輔助處理器」。否則，DCM 將不提供用於選取儲存體選項的頁，作爲憑證建立處理的一部分。

配置步驟

Sam 需要在其公司的 iSeries 伺服器上執行下列步驟，以使用加密硬體來保護私密金鑰：

1. 確定已符合此實務的先決條件和假設。
2. 使用 IBM「數位憑證管理程式 (DCM)」，以建立新的數位憑證，或更新現行的數位憑證：
   a. 選取正簽署現行憑證的憑證權限 (CA) 類型。
   b. 選取**硬體**作爲憑證的私密金鑰的儲存體選項。
   c. 選取您想要在其上儲存憑證的私密金鑰的加密硬體裝置。
   d. 選取要使用的公用 CA。

與新的數位憑證相關的私密金鑰現在儲存在指定於「步驟 2.c」的「4758 加密輔助處理器」上。 Sam 現在可以進入其公司的 Web 伺服器的配置並指定使用新建立的憑證。一旦重新啓動 Web 伺服器，它將使用新的憑證。«

加密硬體實務：撰寫 OS/400 應用程式來使用 4758 加密輔助處理器

»

狀況

假設您是大型財務「金融信用合作社」的一位 iSeries 伺服器程式設計師。已指派您負責處理「IBM 4758-023 加密輔助處理器」PCI 卡 (「硬體特性」碼 4801 或 4802，此後亦稱爲「4758 加密輔助處理器」) 的作業，該卡安裝於「金融信用合作社」iSeries 伺服器中，當於自動提款機 (ATM) 上輸入成員的財務個人識別碼 (PIN) 時驗證它們。

您決定使用「選項 35」的組件 CCA CSP (密碼服務提供程式) API 撰寫 iSeries 伺服器 OS/400 應用程式，以於「4758 輔助處理器」中存取加密服務來驗證成員的個人識別碼 (PIN)。爲「4758 加密輔助處理器」撰寫的 iSeries 伺服器 OS/400 應用程式利用輔助處理器，以執行對安全性敏感的作業和加密作業。

註: 透過 CCA CSP 最多能使用八個「4758 加密輔助處理器」。應用程式必須藉由使用 Cryptographic_Resource_Allocate (CSUACRA) 和 Cryptographic_Resource_Deallocate (CSUACRD) CCA API 來控制對個別「輔助處理器」的存取。

明細

1. 「金融信用合作社」成員在 ATM 上輸入他們的個人識別碼 (PIN)。
2. 在 ATM 上加密個人識別碼 (PIN)，然後透過網路傳送至「金融信用合作社」的 iSeries 伺服器。
3. iSeries 伺服器辨識交易要求，並呼叫程式來驗證成員的個人識別碼 (PIN)。
4. 程式傳送包含加密的個人識別碼 (PIN)、成員的帳戶號碼、個人識別碼 (PIN) 產生金鑰和個人識別碼 (PIN) 加密金鑰的要求至「4758 加密輔助處理器」。
5. 「4758 加密輔助處理器」確認或拒絕個人識別碼 (PIN) 的有效性。
6. 程式傳送「4758 加密輔助處理器」的結果至 ATM。
   a. 若已確認個人識別碼 (PIN)，則該成員可以順利地完成與「金融信用合作社」的交易。
   b. 若個人識別碼 (PIN) 被拒絕，則該成員無法完成與「金融信用合作社」的交易。

先決條件與假設

1. 您公司有一台已適當地安裝並配置有「4758 加密輔助處理器」的 iSeries 伺服器。請參照下列資訊：
   a. 4758 加密輔助處理器的計畫
   b. 配置 4758 加密輔助處理器
   c. 配置 4758 加密輔助處理器以使用 OS/400 應用程式
2. 您熟悉「選項 35：共同密碼架構密碼服務提供程式 (CCA CSP)」。將它套裝爲 OS/400「選項 35」，並且提供安全性應用程式設計介面 (SAPI)，以使您能撰寫允許存取「4758 加密輔助處理器」的加密服務的應用程式。
3. 您有對 CCA Basic Services Guide 之存取權限，其中您可以找到「財務服務支援」動詞以在應用程式中使用。

配置步驟

完成使用「4758 加密輔助處理器」驗證個人識別碼 (PIN) 的目標的一種方法爲，撰寫兩個 iSeries 伺服器 OS/400 應用程式：

1. 撰寫載入個人識別碼 (PIN) 驗證金鑰及個人識別碼 (PIN) 加密金鑰的程式，並將它們儲存於金鑰儲存檔案中。假設已使用未加密金鑰部份，您需要使用下列 API：
   - Logon_Control (CSUALCT)
   - Key_Part_Import (CSNBKPI)
   - Key_Token_Build (CSNBKTB)

- Key_Record_Create (CSNBKRC)
- Key_Record_Write (CSNBKRW)
- 可選用的 API：KeyStore_Designate (CSUAKSD)

2. 撰寫另一個稱爲 Encrypted_PIN_Verify (CSNBPVR) API 的程式以驗證加密個人識別碼 (PIN)，然後將它們有效或無效狀態回報至 ATM。

相關的頁面：配置 4758 加密輔助處理器以與 OS/400 應用程式一起使用

4758 加密輔助處理器的計畫

下列資訊與那些在 iSeries 伺服器上規劃安裝「4758 加密輔助處理器」有關：

- 4758 加密輔助處理器的基本要求
- 對 4758 加密輔助處理器的安全存取權限
- SAPI 所需要的物件權限

4758 加密輔助處理器的基本要求

安裝和使用「4758 輔助處理器」之前，您的伺服器必須符合這些基本要求。

硬體基本要求

iSeries 伺服器的 「4758-023 輔助處理器」可依據指定特性碼 4801 或 4802 而排序。4801 特性由下列 iSeries 伺服器模型支援：

- 250 和 270 (250 需要 7102 擴充裝置)
- 810、820、825、830、840、870 和 890
- SB2 和 SB3
- 擴充裝置 5074、5075、5078、5079、5088、5094、5095 和 5294

您的「4758 輔助處理器」爲一張 PCI 卡。按「輔助處理器」的安裝手冊安裝該卡。

註：若允許置於攝氏 -15 度 (華氏 5 度) 以下冷卻， 「4758 輔助處理器」將破壞其原廠驗證。若「4758 輔助處理器」破壞其原廠驗證，則您不能再使用該卡。關於訂購新的卡，請聯絡硬體服務提供者。

軟體基本要求

「4758 輔助處理器」需要下列軟體：

- OS/400 (5722-SS1)：「4758-023 輔助處理器」(iSeries 伺服器特性碼 4801 或 4802) 需要「OS/400 版本 4 版次 5 修訂 0」或更新版本。
- OS/400 選項 35 共同密碼架構密碼服務提供程式 (CCA CSP)
- OS/400 選項 34 數位憑證管理程式 (若您正規劃使用「4758 輔助處理器」配置 Web 型公用程式)
- OS/400 57xx-TC1 TCP/IP 連通性公用程式 (若您正規劃使用「4758 輔助處理器」配置 Web 型公用程式)
- OS/400 57xx-DG1 IBM HTTP Server (若您正規劃使用「4758 輔助處理器」配置 Web 型公用程式)

- iSeries 伺服器的 128 位元「密碼存取提供者 (5722-AC3) 授權程式產品必須安裝於您的 iSeries 伺服器上，以啓用「4758 加密輔助處理器」的加密功能。此選項啓用您的「4758 輔助處理器」來使用 56 位元 DES 金鑰、112 位元「三重 DES 演算法」金鑰，及 2048 位元 RSA 金鑰。

註:

1. OS/400 版本 5 版次 2 修訂 0，包含了新的「4758 輔助處理器」配置 Web 型公用程式。
2. 在 OS/400 版本 4 版次 5 修訂 0 上用 SSL 使用「4758-023 輔助處理器」之前，必須安裝特殊且可用性有限的 PTF。在 OS/400 版本 5 版次 2 修訂 0 或 OS/400 的更高版次上，以 SSL 使用「4758-023 輔助處理器」不需要特殊的 PTF 。
3. 您也許已安裝了「密碼存取提供者」的前一版 (例如 5769-AC1、5769-AC2、5769-AC3)。這些產品與版本 5 版次 2 修訂 0 相容。

對 4758 加密輔助處理器的安全存取

存取控制只允許讓那些已授權和資源相互作用的使用者使用系統資源。伺服器可讓您控制系統資源的使用者授權。您的組織應該識別組織的安全性階層中的每一個系統資源。該階層應該清楚地記載使用者所具有的資源存取授權等級。

OS/400 選項 35 中的所有服務程式以 *PUBLIC 的 *EXCLUDE 權限出貨。您必須提供使用者所需使用的服務程式之 *USE 權限。此外，您亦必須提供使用者檔案庫 QCCA 中的 QC6SRV 服務程式之 *USE 權限。

參與設置「4758 輔助處理器」的使用者必須具有使用 Master_Key_Process (CSNBMKP)、Access_Control_Initialize (CSUAACI) 或 Cryptographic_Facility_Control (CSUACFC) 安全性應用程式設計介面 (SAPI) 的 *IOSYSCFG 特殊權限。您可以使用這三個 SAPI 來執行「4758 輔助處理器」的所有配置步驟。對所有 SAPI 而言，使用者可能需要附加的物件權限。請參照第 18 頁的『SAPI 所需的物件權限』。

對大部份安全環境而言，考慮將 4758 管理者角色指派給一組沒有 *ALLOBJ 特殊權限的使用者。這樣的話，具有 *ALLOBJ 特殊權限的使用者將無法登入 4758 上的管理角色，所以他們無法改變「輔助處理器」的配置。然而，他們可以控制 SAPI 服務程式的物件權限，防止 4758 管理者的誤用。

爲了使用「4758 加密輔助處理器」配置 Web 型公用程式，使用者必須有 *SECADM 特殊權限。

「4758 輔助處理器」有不同的存取控制，這些存取控制與伺服器的存取控制不相關。「4758 輔助處理器」存取控制可讓您控制對「4758 輔助處理器」硬體指令的存取權限。如需這些指令的相關資訊，請參閱第 22 頁的『建立及定義角色和設定檔』。

如需更高的安全性，可限制「4758 輔助處理器」中預設角色的功能。在其它角色中指派功能，以要求兩人或兩人以上來執行安全性敏感的功能，如變更主要金鑰。當您使用說明於第 22 頁的『建立及定義角色和設定檔』中的角色及設定檔時，可以執行此作業。

註: 您也應該考慮某些標準實體安全性措施，例如將您的伺服器保存於已鎖的門後。

SAPI 所需的物件權限

SAPI	裝置的 *USE	DES 金鑰庫檔案的 *USE	DES 金鑰庫檔案的 *CHANGE	「DES 金鑰庫檔案庫」的 *USE	PKA 金鑰庫檔案的 *USE	PKA 金鑰庫檔案的 *CHANGE	「PKA 金鑰庫檔案庫」的 *USE
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]
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			

SAPI	裝置的 *USE	DES 金鑰庫檔案的 *USE	DES 金鑰庫檔案的 *CHANGE	「DES 金鑰庫檔案庫」的 *USE	PKA 金鑰庫檔案的 *USE	PKA 金鑰庫檔案的 *CHANGE	「PKA 金鑰庫檔案庫」的 *USE
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						
CSUACFQ	Y						
CSUACRA	Y						
CSUACRD	Y						
CSUAKSD							
CSUALCT	Y						
CSUAMKD	Y						

[1]可選擇使用此 API 的「資料加密標準 (DES)」或公開金鑰演算法 (PKA) 金鑰庫檔案。

[2]多個參數可以選用性地使用金鑰庫檔案。那些參數中，每一個的權限基本要求不同。

[3]Key_Store_Initialize SAPI 不會同時要求對兩個檔案的權限。

[4]這些 SAPI 僅專屬於「4758-023 輔助處理器」。

配置 4758 加密輔助處理器

配置「4758 輔助處理器」，可讓您開始使用它所有的加密作業。

配置「4758 輔助處理器」最容易最快捷的方法是使用「4758 加密輔助處理器」配置 Web 型公用程式，可在位於 http://*server-name*:2001 的 iSeries 伺服器「作業」頁找到它。公用程式包括用於配置 (及起始設定) 先前尚未配置之「輔助處理器」的「基本」配置精靈。如果 HTTP 和 SSL 先前尚未配置，在您使用「配置精靈」之前，將需要做下列操作。

- 啓動 HTTP 管理伺服器。
- 配置 HTTP 管理伺服器以使用 SSL。
- 使用 DCM 來建立憑證，指定此私密金鑰產生並儲存於軟體中。
- 使用 DCM 來接收已簽署的憑證。
- 將憑證與 HTTP 管理伺服器應用程式 ID 結合。
- 重新啓動 HTTP 管理伺服器，以爲 SSL 處理程序啓用它。

如果已配置了「4758 輔助處理器」，則在**管理配置**選項上按一下，以變更「輔助處理器」的特定部份的配置。

如果您偏好撰寫自己的應用程式來配置「輔助處理器」，您可透過使用 Cryptographic_Facility_Control (CSUACFC)、Access_Control_Initialize (CSUAACI)、Master_Key_Process (CSNBMKP) 及 Key_Store_Initialize (CSNBKSI) API 動詞來實現。此節中的許多頁包括一個或多個程式範例，顯示如何透過應用程式配置「輔助處理器」。變更這些程式以滿足您的特定需要。

無論您選擇使用「4758 加密輔助處理器」配置公用程式，還是撰寫自己的應用程式，下列內容略述了要適當配置「4758 輔助處理器」必須採取的步驟：

1. 第 21 頁的『建立裝置說明』. 裝置說明會指定金鑰儲存體的預設位置。無論是命名還是不命名金鑰儲存檔案，都可建立裝置說明。
2. 第 21 頁的『爲金鑰儲存檔之檔案命名』. 在您可以使用金鑰儲存檔案或儲存在金鑰儲存檔案中的金鑰執行任何作業之前，必須先命名此金鑰儲存檔案。您可以使用程式來明確地命名金鑰儲存檔案，也可以在裝置說明上命名它們。您可以命名一個檔案來儲存「資料加密標準 (DES)」及「三重 DES 演算法」金鑰，並命名另一個檔案來儲存公開金鑰演算法 (PKA) 金鑰。透過命名儲存金鑰的檔案，您可以設定該資料庫以包含 DES (及「三重 DES 演算法」) 及 PKA 金鑰。如果要在自己的金鑰儲存檔案中保存金鑰，您應該使用程式來命名金鑰儲存檔案。如果您不用程式命名金鑰儲存檔案， CCA CSP 會將金鑰儲存於裝置說明上命名的金鑰儲存檔案中。
3. 第 22 頁的『建立及定義角色和設定檔』. 將使用者指派到這些角色及設定檔時，您要決定「4758 輔助處理器」將允許使用者使用什麼加密函數。
4. 第 59 頁的『設定環境 ID 與時鐘』. 「4758 輔助處理器」使用 EID 來驗證哪個「4758 輔助處理器」建立金鑰記號。它使用時鐘來取得時間與日期戳記，並控制設定檔是否可登入。
5. 第 70 頁的『載入函數控制向量』. 函數控制向量會告訴「4758 輔助處理器」使用多大金鑰長度來建立金鑰。如果不載入函數控制向量，則您無法執行任何加密功能。
6. 第 82 頁的『載入及設定主要金鑰』. 於載入函數控制向量後，請載入並設定主要金鑰。您可使用主要金鑰加密其它金鑰。

建立裝置說明

您必須爲伺服器上的「4758 輔助處理器」建立裝置說明。由 CCA CSP 使用裝置說明，可協助向「4758 輔助處理器」發出加密要求。此外，裝置說明爲「4758 輔助處理器」提供了金鑰儲存檔案儲存體的預設位置。 「4758 加密輔助處理器」配置公用程式中的「基本」配置精靈 (可於 http://*server-name*:2001 處的 iSeries 伺服器「作業」頁中找到)，能爲您建立裝置說明，或者您也可以自己使用 Create Device Crypto CL 指令建立裝置說明。

若要使用「基本」配置精靈來建立裝置說明，請遵循下列步驟：

1. 請將您的 Web 瀏覽器指向該 iSeries 伺服器「作業」頁：`http://server-name:2001`
2. 按一下「4758 加密輔助處理器」配置。
3. 按一下標籤爲**啓動安全階段作業**的按鈕。
4. 按一下**基本配置**精靈。
5. 在**歡迎**頁面上按一下**繼續**。
6. 爲要使用的資源，按一下裝置名稱設定爲 `*CREATE` 的清單項目。
7. 正如「基本」配置精靈所指示的繼續。

若要使用 CL 指令建立裝置說明，請遵循下列步驟：

1. 在 CL 指令行上鍵入 CRTDEVCRP。
2. 依據提示指定裝置名稱。若要設定預設裝置，請命名裝置 CRP01。否則，爲了存取裝置說明，您建立的每一個應用程式都必須使用「加密資源配置 (CSUACRA) API」。
3. 指定預設 PKA 金鑰儲存檔的名稱，或將參數預設爲 `*NONE`。
4. 指定預設 DES 金鑰儲存檔的名稱，或將參數預設爲 `*NONE`。
5. 依據提示指定說明。此爲可選用的。
6. 一旦建立了裝置說明，就能使用 Vary Configuration (VRYCFG) 或 Work with ConfigurationStatus (WRKCFGSTS) CL 指令來轉接裝置。
7. 此通常需要一分鐘，但也可能需要十分鐘才能完成。

現在您已完成了裝置說明的建立。

為金鑰儲存檔之檔案命名

於使用金鑰儲存檔或儲存於金鑰儲存檔中的金鑰執行任何作業之前，必須命名金鑰檔。這將使「4758 輔助處理器」指向正確的檔案。您可以命名兩種類型的金鑰儲存檔。一種類型儲存「資料加密標準 (DES)」金鑰和「三重 DES 演算法」金鑰。DES 和「三重 DES 演算法」是對稱的加密演算法；「4758 輔助處理器」使用相同的金鑰來加密和解密。另一種類型儲存公開金鑰演算法 (PKA) 金鑰。公開金鑰演算法爲不對稱的；金鑰成對產生。「4758 輔助處理器」使用一個金鑰加密，使用另一個解密。「4758 輔助處理器」支援 RSA 公開金鑰演算法。

可以使用程式明確地命名金鑰儲存檔，或者也可以藉由在裝置說明上配置它來命名該檔。如果要從程式命名金鑰儲存檔，請使用 Key_Store_Designate (CSUKSD) 安全性應用程式設計介面 (SAPI)。如果使用程式命名金鑰儲存檔，那麼「4758 輔助處理器」僅使用執行程式之工作的名稱。然而，藉由在程式中明確地命名金鑰儲存檔，可以使用區別於其他使用者的金鑰儲存檔。如果在裝置說明上命名了金鑰儲存檔，那麼就不需

要在程式中命名它們。如果您正嘗試維護跨多重 IBM 平台的相同程式來源檔，那麼這會對您有所協助。如果您正從「共同密碼架構」的另一施行轉向某個程式，那麼此亦有用。

需要將加密金鑰儲存在一個安全套表中，這樣您就能隨時使用它們，並與其他使用者和伺服器交換它們。可以使用您自己的方法來儲存加密金鑰，或將它們儲存在金鑰儲存檔中。您可以任意擁有多個金鑰儲存檔，還可以爲每一個類型的金鑰建立多重金鑰儲存檔。可以在您的金鑰儲存檔中任意放置多個加密金鑰。

因爲每一個金鑰儲存檔都是單獨的伺服器物件，所以可以爲每一個檔案授權不同的使用者。可以分次儲存和復置每一個金鑰儲存檔。這視檔案資料變更的頻率或者它正保護哪個資料而定。

建立及定義角色和設定檔

「4758 輔助處理器」使用角色型的存取控制。在角色型的系統中，您可以定義一組對應於「4758 輔助處理器」使用者類別的角色。可以藉由定義相關的使用者設定檔，將使用者對映於一個有效角色，來列名每一個使用者。

角色功能視存取控制點，或已爲該角色啓用的加密硬體指令而定。然後可以使用「4758 輔助處理器」來建立依據所選擇之角色的設定檔。

一個角色型的系統比一個爲每一個使用者個別分派權限的系統更有效。一般而言，可以將使用者區分爲一些具有不同存取權的種類。角色的使用可讓您在角色套表中，定義每一個種類一次。

角色型存取控制系統及可以使用之容許的指令分組，均被設計用來支援各種安全原則。特別是您可以設定「4758 輔助處理器」，來執行雙重控制，分割知識原則。依據此原則，一旦完全啓動了「4758 輔助處理器」，任何人都無法導致除拒絕服務攻擊之外的有害動作。如果要執行此原則及其它許多方法，您需要限制某些指令的使用。設計應用程式時，請考慮必須在存取控制系統中啓用或限制的指令及相關的安全原則。

每一個「4758 輔助處理器」必須有一個稱爲預設角色的角色。任何沒有登入到「4758 輔助處理器」的使用者，都將操作定義於預設角色中的功能。僅需要定義於預設角色中功能的使用者，不需要設定檔。在大部份應用程式中，大多數使用者將使用預設角色操作，且沒有使用者設定檔。通常，僅有安全主管及其他特殊使用者需要設定檔。

「4758 輔助處理器」處於未起始設定狀態時，預設角色會啓用下列存取控制點：

- PKA96 單向雜湊
- 設定時鐘
- 重新起始設定裝置
- 起始設定存取控制系統角色及設定檔
- 變更使用者設定檔中的期滿資料
- 重設使用者設定檔中的登入失敗計數
- 讀取公用存取控制資訊
- 刪除使用者設定檔
- 刪除角色

預設角色在最初定義，以使允許的功能會是那些與存取控制起始設定相關的功能。此保證了於執行任何有用的加密工作前，會起始設定「4758 輔助處理器」。該基本要求防止了將「4758 輔助處理器」放置在服務程式中時，某人可能偶然地保留了完整權限，這樣的安全性「事故」。

定義角色

定義新角色 (及重新定義預設角色) 的最容易和最快的方法是，使用可在 http://*server-name*:2001 處的 iSeries 伺服器「作業」頁中找到的「4758 加密輔助處理器」配置 Web 型公用程式。公用程式包括當「輔助處理器」處於未起始設定狀態時所使用的「基本」配置精靈。「基本」配置精靈，可以與重新定義預設角色一起，定義一或三個管理角色。如果已起始設定了「4758 輔助處理器」，請按一下**管理配置**，然後按一下**角色**以定義新建角色，或變更或刪除現有的角色。

如果您偏好撰寫自己的應用程式來管理角色，那麼您可以使用 Access_Control_Initialization (CSUAACI) 及 Access_Control_Maintenance (CSUAACM) API 動詞。若要變更「4758 輔助處理器」中預設角色，請以 ASCII 編碼 ″DEFAULT″ 指定爲適當的參數。您必須將一個 ASCII 空格字元填充於此。否則，可能會在使用於角色 ID 或設定檔 ID 的字元上沒有限制。提供四個範例程式以作爲您的參考。其中兩個範例以 ILE C 撰寫，而另兩個範例以 ILE RPG 撰寫。兩組範例執行相同的功能。

- 第 26 頁的『範例：爲 4758 輔助處理器建立角色和設定檔的 ILE C 程式』
- 第 46 頁的『範例：爲 4758 輔助處理器啓用所有預設角色中的存取控制點之 ILE C 程式』
- 第 37 頁的『範例：爲 4758 輔助處理器建立角色或設定檔的 ILE RPG 程式』
- 第 51 頁的『範例：爲 4758 輔助處理器之預設角色啓用所有存取控制點的 ILE RPG 程式』

註: 如果您選擇使用提供的程式範例之一，則請變更它以滿足您的特定需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

定義設定檔

爲「4758 輔助處理器」建立及定義角色後，可以建立一個此角色所使用的設定檔。設定檔允許使用者存取「4758 輔助處理器」中，對於預設角色可能未啓用的特定功能。

定義新設定檔最容易和最快的方法是，使用於 http://*server-name*:2001 處的 iSeries 伺服器「作業」頁中找到的 「4758 加密輔助處理器」配置 Web 型公用程式。公用程式包括當「輔助處理器」處於未起始設定狀態時所使用的「基本」配置精靈。「基本」配置精靈可以定義一或三個管理設定檔。如果已起始設定「4758 輔助處理器」，請按一下**管理配置-->設定檔** 以定義新建設定檔，或變更或刪除現有的設定檔。

如果要撰寫自己的應用程式來管理設定檔，那麼您可以使用 Access_Control_Initialization (CSUAACI) 及 Access_Control_Maintenance (CSUAACM) API 動詞。爲您提供了兩個範例程式：

- 第 55 頁的『範例：爲 4758 輔助處理器變更現有設定檔的 ILE C 程式』
- 第 57 頁的『範例：變更 4758 輔助處理器現有設定檔的 ILE RPG 程式』

註: 如果您選擇使用提供的程式範例之一，則請變更它以滿足您的特定需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

如果您要將「4758 輔助處理器」使用於 SSL，那麼必須至少爲預設角色授權下列存取控制點：

- 數位簽章建立
- 數位簽章驗證
- PKA 金鑰建立
- PKA 複製金鑰建立
- RSA 加密清除資料
- RSA 解密清除資料
- 刪除保留金鑰
- 列出保留金鑰

「4758 加密輔助處理器」配置公用程式中的「基本」配置精靈會自動重新定義預設角色，以使它無需任何變更即可使用於 SSL。

設定所有角色及設定檔之後，爲了避免安全性事故，請考慮爲該預設角色拒絕下列存取控制點 (也稱爲加密硬體指令)：

註: 您應該僅啓用那些正常作業必需的存取控制點。在最大值時，您應該僅啓用特別必要的功能。若要決定哪個存取控制點是必要的，請參照 CCA Basic Services Guide。每一個 API 列出該 API 所必要的存取控制點。若您不需要使用特殊的 API，請考慮停用其必要的存取控制點。

- 載入「主要金鑰」的第一個組件
- 結合主要金鑰組件
- 設定主要金鑰
- 建立隨機主要金鑰
- 清除新的主要金鑰登記
- 清除舊的主要金鑰登記
- 轉換 CV
- 設定時鐘

  警告: 若您想從預設的角色停用「設定時鐘」存取控制點，請確定您停用存取之前時鐘已設定。當使用者嘗試登入時，時鐘會爲「4758 輔助處理器」所使用。若時鐘設定錯誤，則使用者無法登入。
- 重新起始設定裝置
- 起始設定存取控制系統
- 變更鑑別資料 (例如，通關密語)
- 重設密碼失效計數
- 讀取公用存取控制資訊
- 刪除使用者設定檔
- 刪除角色
- 載入函數控制向量
- 清除函數控制向量
- 強制使用者登出
- 設定 EID

- 起始設定主要金鑰複製控制
- 登記公開金鑰雜湊
- 登記公開金鑰並複製
- 登記公開金鑰
- PKA 複製金鑰建立 (SSL 所必要的存取控制點)
- 複製資訊獲得組件 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15
- 複製資訊安裝組件 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15
- 刪除保留的金鑰 (SSL 所必要的存取控制點)
- 列出保留的金鑰 (SSL 所必要的存取控制點)
- 在主要金鑰下加密
- 資料金鑰匯出
- 資料金鑰匯入
- 對主要金鑰重新加密
- 從主要金鑰重新加密
- 載入第一個金鑰組件
- 結合金鑰組件
- 新增金鑰組件
- 完成金鑰組件

對大部份安全環境而言，考慮在起始設定存取控制系統之後將其鎖定。您可以藉由刪除任何允許使用「存取控制起始設定」或「刪除角色」存取控制點的設定檔而使存取控制系統變爲不可變更。若無這些存取控制點，則任何角色的進一步變更是不可能的。具有使用「起始設定存取控制」或「刪除角色」存取控制點的權限，就可以刪除 DEFAULT 角色。

刪除 DEFAULT 角色將導致起始 DEFAULT 角色自動重建。起始 DEFAULT 角色允許設定任何功能。對這些存取控制點有存取權限的使用者，對存取控制系統的整個操作，擁有無限制的權限。於「4758 輔助處理器」進入正常作業前，透過使用 Access_Control_Maintenance (CSUAACM) 和 Cryptographic_Facility_Query (CSUACFQ) API 動詞，可以審核存取控制設定。

若因爲某種理由而使狀態回應與預期的不一樣，則「4758 輔助處理器」不應該用於應用程式目的，直至它已重新配置符合您的安全原則。若角色包含變更密碼字詞的許可權，則任何設定檔的密碼字詞都可以變更。您應考慮是否應該允許密碼字詞變更及如果允許，以及哪個角色應該有此權限。

若有使用者報告無法登入，則應該報告至具有通關密語變更許可權之個人之外的其他人 (或是同時報告該人員及其他人員)。考慮定義角色，以使雙重控制對於每一個安全性敏感的作業是必要的，以防止內部人員自己的惡意行爲。例如，考慮在兩個或更多的角色間分割下列存取控制點群組。建議一個人不應該具有使用「主要」金鑰群組中的所有指令的能力，因爲這可能會有安全性風險。

「主要」金鑰群組由這些存取控制點組成：

- 載入「主要金鑰」的第一個組件
- 結合主要金鑰組件

- 設定主要金鑰
- 建立隨機主要金鑰
- 清除新的主要金鑰登記
- 清除舊的主要金鑰登記

同樣的記號，不應該將「複製」金鑰群組中的所有指令授權給一個人。

「複製」金鑰群組由這些存取控制點組成：
- 起始設定主要金鑰複製控制
- 登記公開金鑰雜湊
- 登記公開金鑰並複製
- 登記公開金鑰
- PKA 複製金鑰建立
- 複製資訊獲得組件 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15
- 複製資訊安裝組件 1、2、3、4、5、6、7、8、9、10、11、12、13、14、15

您爲「4758 輔助處理器」建立和定義設定檔之後，必須依第 70 頁的『載入函數控制向量』中說明，爲「4758 輔助處理器」載入函數控制向量。若無函數控制向量，您的「4758 輔助處理器」無法執行任何加密功能。

範例：為 4758 輔助處理器建立角色和設定檔的 ILE C 程式

變更此程式範例以滿足您爲「4758 輔助處理器」建立角色或設定檔的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*--------------------------------------------------------------------*/
/*  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");
}
```

範例：為 4758 輔助處理器建立角色或設定檔的 ILE RPG 程式

變更此程式範例，以滿足爲「4758 輔助處理器」建立角色和設定檔的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

範例：為 4758 輔助處理器啓用所有預設角色中的存取控制點之 ILE C 程式

變更此程式範例，以滿足您爲「4758 輔助處理器」啓用所有預設角色中控制點的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
/*-------------------------------------------------------------------*/
/*  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");
}
```

範例：為 4758 輔助處理器之預設角色啓用所有存取控制點的 ILE RPG 程式

變更此程式範例，以滿足您爲「4758 輔助處理器」啓用所有預設角色中控制點的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
     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.
```

範例：為 4758 輔助處理器變更現有設定檔的 ILE C 程式

變更此程式範例，以滿足您變更「4758 輔助處理器」現有設定檔的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*------------------------------------------------------------------*/
/* 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);
   }
}
```

範例：變更 4758 輔助處理器現有設定檔的 ILE RPG 程式

變更此程式範例，以滿足您變更「4758 輔助處理器」現有設定檔的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

設定環境 ID 與時鐘

環境 ID (EID)

「4758 輔助處理器」儲存 EID 爲 ID。設定 EID 的最容易最快捷的途徑是使用「4758 加密輔助處理器」配置 Web 型公用程式，可在位於 http://*server-name*:2001 的 iSeries

「作業」頁找到它。公用程式包括當「輔助處理器」處於未起始設定狀態時所使用的「基本」配置精靈。如果已起始設定「4758 輔助處理器」，則在**管理配置**上按一下，然後在**屬性**上按一下，以設定 EID。

如果您偏好撰寫自己的應用程式來設定 EID，可以藉由使用 Cryptographic_Facility_Control (CSUACFC) API 動詞來這樣做。 提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 『範例：於 4758 輔助處理器上設定環境 ID 的 ILE C 程式』
- 第 62 頁的『範例：於 4758 輔助處理器上設定環境 ID 的 ILE RPG 程式』

「4758 輔助處理器」將 EID 複製到每一個「4758 輔助處理器」建立的 PKA 金鑰記號中。EID 會協助「4758 輔助處理器」識別由它建立的，而與由另一個「4758 輔助處理器」建立的金鑰相對立的金鑰。

時鐘

「4758 輔助處理器」使用它的時鐘日曆來記錄日期與時間，並確定設定檔是否可以登入。預設時間爲「格林威治標準時間 (GMT)」。因爲它的功能，所以在移除預設角色之設定它的能力前，您應當設定「4758 輔助處理器」中的時鐘。

設定時鐘的最容易最快捷的途徑是使用「4758 加密輔助處理器」配置 Web 型公用程式，可在位於 http://*server-name*:2001 的 iSeries「作業」頁找到它。公用程式包括當「輔助處理器」處於未起始設定狀態時所使用的「基本」配置精靈。如果已起始設定「4758 輔助處理器」，則在**管理配置**上按一下，然後在**屬性**上按一下，以設定時鐘。

如果您偏好撰寫自己的應用程式來設定時鐘，可以藉由使用 Cryptographic_Facility_Control (CSUACFC) API 動詞來這樣做。 提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 第 65 頁的『範例：於 4758 輔助處理器上設定時鐘的 ILE C 程式』
- 第 67 頁的『範例：於 4758 輔助處理器上設定時鐘的 ILE RPG 程式』

範例：於 4758 輔助處理器上設定環境 ID 的 ILE C 程式

變更此程式範例，以滿足您在「4758 輔助處理器」上設定環境 ID 的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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);
   }
}
```

範例：於 4758 輔助處理器上設定環境 ID 的 ILE RPG 程式

變更此程式範例，以滿足您在「4758 輔助處理器」上設定環境 ID 的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

範例：於 4758 輔助處理器上設定時鐘的 ILE C 程式

變更此程式範例，以滿足於「4758 輔助處理器」上設定時鐘的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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);
    }
}
```

範例：於 4758 輔助處理器上設定時鐘的 ILE RPG 程式

變更此程式範例，以滿足於「4758 輔助處理器」上設定時鐘的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

載入函數控制向量

於第 22 頁的『建立及定義角色和設定檔』後，必須載入「4758 輔助處理器」的函數控制向量 (FCV)。沒有它，「4758 輔助處理器」將無法執行任何加密作業。

函數控制向量是一個儲存於 IBM 所提供檔案中之已數位簽章的值。當您安裝「5722-ACx 密碼存取提供者」產品之一時，該檔案已儲存於根檔案系統中，路徑爲 /QIBM/ProdData/CAP/FCV.CRT。該值會在「4758 輔助處理器」內啓用加密應用程式，以產生與適用的匯入匯出規則一致的加密服務層次。

載入 FCV 最簡單最便捷的方法就是使用「4758 加密輔助處理器」配置 Web 型公用程式，它可以在 http://*server-name*:2001 的 iSeries「作業」頁中找到。公用程式包括當「輔助處理器」處於未起始設定狀態時所使用的「基本」配置精靈。若已起始設定「4758 輔助處理器」，則請按一下**管理配置**，然後按一下**屬性**以載入 FCV。

若您偏好撰寫自己的應用程式來載入 FCV，可以藉由使用 Cryptographic_Facility_Control (CSUACFC) API 動詞來這樣做。提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 第 71 頁的『範例：爲 4758 輔助處理器載入函數控制向量的 ILE C 程式』
- 第 73 頁的『範例：爲 4758 輔助處理器載入函數控制向量的 ILE RPG 程式』

提供另外兩個範例程式，以顯示如何清除函數控制向量。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。

- 第 78 頁的『範例：由 4758 輔助處理器清除函數控制向量的 ILE C 程式』
- 第 79 頁的『範例：由 4758 輔助處理器清除函數控制向量的 ILE RPG 程式』

在爲「4758 輔助處理器」載入函數控制向量後，您可以使用用於加密金鑰的第 82 頁的『載入及設定主要金鑰』來載入及設定主要金鑰。

範例：為 4758 輔助處理器載入函數控制向量的 ILE C 程式

請變更此程式範例，以滿足您爲「4758 輔助處理器」載入函數控制向量的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------*/
/* 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;
  }
 }
```

範例：為 4758 輔助處理器載入函數控制向量的 ILE RPG 程式

請變更此程式範例，以滿足您爲「4758 輔助處理器」載入函數控制向量的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

範例：由 4758 輔助處理器清除函數控制向量的 ILE C 程式

變更此程式範例，以滿足由「4758 輔助處理器」清除函數控制向量的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------*/
/* 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");
 }
```

範例：由 4758 輔助處理器清除函數控制向量的 ILE RPG 程式

變更此程式範例，以滿足由「4758 輔助處理器」清除函數控制向量的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

載入及設定主要金鑰

於第 70 頁的『載入函數控制向量』後，您可以載入及設定主要金鑰。「4758 輔助處理器」使用主要金鑰來加密所有作業金鑰。主要金鑰爲儲存在「4758 輔助處理器」安全模組內的清除 (未加密) 之特殊的加密用的金鑰。 「4758 輔助處理器」使用主要金鑰來加密其它金鑰，這樣您可以在「4758 輔助處理器」外部儲存那些金鑰。主要金鑰爲至少兩個 168 位元的組件互斥 OR 所組成之 168 位元的金鑰。

載入主要金鑰

主要金鑰有三種註冊：「新的」、「現行的」及「舊的」。新的主要金鑰註冊用於保留在建立時被擱置的主要金鑰。它不用於加密任何金鑰。「現行的」主要金鑰註冊會保留當前用於加密新產生的/匯入的/重新加密的金鑰之主要金鑰。舊的主要金鑰註冊保留前一個主要金鑰。它用於在主要金鑰變更發生之後回復金鑰。當您載入主要金鑰時，「4758 輔助處理器」將它放入「新的」主要金鑰註冊。在您設定主要金鑰之前，它都會保留在那裡。

根據您的安全性需要，選擇這三種方式中的一種來建立及載入主要金鑰：

- 分別載入第一個金鑰組件及後續的金鑰組件，以將金鑰的分割資訊作爲一個整體來維護。這是最不安全的方法，但是您可以藉由爲單獨個人提供每一個金鑰組件來增加安全性。
- 使用隨機金鑰產生，移除金鑰的任何人類資訊。這是載入主要金鑰的最安全的方法，但是您需要將此隨機產生的主要金鑰複製到其它「4758 輔助處理器」以擁有它的複本。
- 藉由從另一個「輔助處理器」複製預先存在的主要金鑰來使用它。

如需複製主要金鑰的相關資訊，請參閱 http://www.ibm.com/security/cryptocards/html/library.shtml。

設定主要金鑰

設定主要金鑰會導致位於「現行的」主要金鑰註冊中的金鑰移動到「舊的」主要金鑰註冊中。然後，位於「新的」主要金鑰註冊中的主要金鑰會移動到「現行的」主要金鑰註冊中。

註: 對於擷取由主要金鑰加密的資料而言，您隨時擁有一份主要金鑰的備份是非常重要的。例如，將它寫在紙上，並且請確定您以適當的安全預防措施儲存備份。或者，複製主要金鑰至另一個「輔助處理器」。

載入及設定主要金鑰的最簡單最快捷的方法是使用「4758 加密輔助處理器」配置 Web 型公用程式，它可以在 http://*server-name*:2001 的 iSeries「作業」頁中找到。公用程式包括當「輔助處理器」處於未起始設定狀態時所使用的「基本」配置精靈。若已起始設定「4758 輔助處理器」，則請按一下**管理配置**，然後按一下**主要金鑰**，以載入並設定主要金鑰。

若您偏好撰寫自己的應用程式來載入及設定主要金鑰，可以藉由使用 Master_Key_Process (CSNBMKP) API 動詞來這樣做。 提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 『範例：載入主要金鑰至「4758 輔助處理器」的 ILE C 程式』
- 第 85 頁的『範例：載入主要金鑰至「4758 輔助處理器」的 ILE RPG 程式』

註: 如果您選擇使用提供的程式範例之一，則請變更它以滿足您的特定需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

重新加密金鑰

當您設定主要金鑰時，應該重新加密由先前的主要金鑰加密的所有金鑰，以避免喪失對它們的存取權限。您必須於變更及設定主要金鑰之前這樣做。

您可以藉由使用「4758 加密輔助處理器」配置 Web 型公用程式來重新加密金鑰儲存檔中的金鑰，該公用程式可以在 http://*server-name*:2001 的 iSeries「作業」頁中找到。必須已起始設定「4758 輔助處理器」。按一下「管理配置」，然後按一下「DES 金鑰」來重新加密 DES 金鑰，或者按一下「PKA 金鑰」來重新加密 PKA 金鑰。

若您擁有不位於金鑰儲存檔中的金鑰，或您偏好撰寫自己的應用程式來重新加密金鑰，可以藉由使用 Key_Token_Change (CSNBKTC) 或 PKA_Key_Token_Change (CSNDKTC) API 動詞來這樣做。提供一個範例程式，供您參考。

- 第 89 頁的『範例：爲「4758 輔助處理器」重新加密金鑰之 ILE C 程式』

註: 如果您選擇使用提供的程式範例，請變更它以滿足特定的需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

範例：載入主要金鑰至「4758 輔助處理器」的 ILE C 程式

請變更此程式範例，以滿足您載入新的主要金鑰至「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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);
    }

}
```

範例：載入主要金鑰至「4758 輔助處理器」的 ILE RPG 程式

請變更此程式範例，以滿足您載入新的主要金鑰至「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

範例：為「4758 輔助處理器」重新加密金鑰之 ILE C 程式

變更此程式範例，以滿足您爲「4758 輔助處理器」重新加密金鑰之需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-----------------------------------------------------------------*/
/* 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() */
```

配置 4758 加密輔助處理器以與 DCM 及 SSL 一起使用

下列章節列出了使「4758-023 輔助處理器」可與 SSL 一起使用所需的步驟。

與 DCM 及 SSL 一起使用「4758-023 輔助處理器」

若要安裝「4758-023 輔助處理器」及必備軟體，您必須執行如下：

- 在伺服器中安裝「輔助處理器」。

  對於特性 4801，依與「4758-023 輔助處理器」一起出貨的「4801 PCI 加密輔助處理器卡指示」中所指示的，來安裝「4758-023 輔助處理器」。

  對於特性 4802，請詢問您的 IBM 硬體客戶服務代表，以安裝「4758-023 輔助處理器」。
- 安裝 OS/400 選項 35 CCA CSP。
- 或者安裝「5722-AC3 密碼存取提供者」128 位元授權程式產品。
- 設定第 17 頁的『對 4758 加密輔助處理器的安全存取』的 OS/400 物件權限。
- 使用 Web 瀏覽器，至 iSeries「作業」頁，該頁爲 http://*server-name*:2001。
- 遵循第 20 頁的『配置 4758 加密輔助處理器』中的步驟來配置 4758。

「4758-023 輔助處理器」現在可用來建立 SSL 憑證的私密金鑰。

- 使用 DCM 來建立憑證，指定此私密金鑰由硬體產生。
- 使用 DCM 來接收已簽署的憑證。

請參閱管理 SSL 通信階段作業的公用網際網路憑證，以取得最後兩個步驟的相關資訊。

註: 如果計畫使用 SSL 的多重卡片，請參閱第 176 頁的『管理多重 4758 加密輔助處理器』及第 186 頁的『複製主要金鑰』。

配置 4758 加密輔助處理器以與 OS/400 應用程式一起使用

下列章節列出能使「4758-023 輔助處理器」可以與 OS/400 應用程式一起使用所需的步驟。請參閱加密硬體實務：撰寫 OS/400 應用程式來使用「4758 加密輔助處理器」，以獲得於 OS/400應用程式中 「4758 加密輔助處理器」的範例用法。

使用 OS/400 應用程式的「4758-023 輔助處理器」

若要安裝「4758 輔助處理器」及必備軟體，您必須進行下列操作：

- 在伺服器中安裝「輔助處理器」。

  對於特性 4801，請依照與「4758 輔助處理器」一起出貨的「4801 PCI 加密輔助處理器卡指令」中的指示來安裝「4758 輔助處理器」。

  對於特性 4802，請詢問您的 IBM 硬體客戶服務代表以安裝「4758 輔助處理器」。
- 安裝 OS/400 選項 35 CCA CSP。
- 安裝「5722-AC3 密碼存取提供者 128 位元」或「5722-AC2 密碼存取提供者 56 位元」的授權程式產品。
- 設定第 17 頁的『對 4758 加密輔助處理器的安全存取』的 OS/400 物件權限。
- 使用 Web 瀏覽器，至 iSeries「作業」頁，該頁爲 http://*server-name*:2001。
- 遵循第 20 頁的『配置 4758 加密輔助處理器』中的步驟來配置 4758。
- 撰寫應用程式以使用「加密輔助處理器」。

註: 若您計畫對 OS/400 應用程式使用多重卡，請參閱第 176 頁的『管理多重 4758 加密輔助處理器』。

移轉至 4758 加密輔助處理器

此資訊解釋如何執行下列移轉：

- 由其它 iSeries 加密產品移轉至「4758 加密輔助處理器」。
- 移轉金鑰儲存檔案

由其它 iSeries 加密產品移轉至「4758 加密輔助處理器」

如果您曾使用過加密，那麼在您的伺服器上可能有兩個加密產品之一。您可以從 iSeries (5799-FRF) 產品的 「IBM CCA 服務」取得金鑰儲存檔案。或者，您可以從 iSeries (5769-CR1) 的「密碼支援」獲得加密交域檔案。如果發生這種情況，您可以將它們的內容移轉至新的「4758 輔助處理器」。每一個加密產品都有一個可用的範例移轉程式：

- iSeries (5799-FRF) 的「**IBM CCA 服務**」。 此產品藉由使用「資料加密標準 (DES)」，提供加密硬體上的加密功能。「共同密碼架構 (CCA) 服務」需要在伺服器上安裝了加密處理器，特性碼爲 2620 或 2628。您可以使用第 95 頁的『從 iSeries 的「 IBM CCA 服務」移轉金鑰儲存檔案』，從「IBM CCA 服務」移轉金鑰儲存檔案至「4758 輔助處理器」。
- **對 iSeries (5769-CR1 或 5722-CR1) 的「密碼支援」**。「密碼支援」僅爲軟體產品，它是在主電腦主要鍵下的加密跨網域金鑰。然後，「密碼支援」會將跨網域金鑰儲存於檔案中。您可以使用第 104 頁的『爲 iSeries 跨網域金鑰檔案移轉「密碼支援」』，從 iSeries 伺服器的「密碼支援」移轉跨網域金鑰至「4758輔助處理器」。

從 iSeries 的「 IBM CCA 服務」移轉金鑰儲存檔案

如果您目前使用 iSeries (5799-FRF) 的「共同密碼架構 (CCA) 服務」，您可以在金鑰儲存檔案中移轉金鑰，以便「4758 輔助處理器」可以使用它們。「輔助處理器」使用具有「CCA 密碼服務提供程式」(CCA CSP，其包裝爲 OS/400 選項 35) 的移轉金鑰。

註: 因爲「CCA 服務」比「4758 輔助處理器」支援的金鑰型類範圍更大，所以您無法移轉所有金鑰。例如，您不能移轉於控制向量集中有禁止匯出位元的金鑰。同樣，您也不能在 iSeries 的「CCA 服務」中移轉任何 PKA 金鑰，因爲「CCA 服務」提供的公開金鑰演算法 (PKA) 支援與在「4758 輔助處理器」中所提供的支援明顯不同。

您將需要撰寫兩個程式，以移轉「資料加密標準 (DES)」金鑰 。或者，有兩個程式範例，『範例：匯出 (EXPORT) 金鑰』及第 100 頁的『範例：匯入 (IMPORT) 金鑰』，您能變更並執行它們以移轉金鑰儲存檔案。 CCA 定義外部 DES 金鑰記號的格式，因此對兩個產品都是相同的。

使用與匯入 (IMPORT) 程式連結的匯出 (EXPORT) 程式。這將從 iSeries 的 「IBM CCA 服務」移轉 DES 金鑰至「4758 輔助處理器」及 CCA CSP。您應該先執行匯出 (EXPORT) 程式以產生檔案，該檔案包含具有安全、可匯出格式的必需的金鑰資訊。然後，您應該轉送檔案至目標伺服器。這樣，您就可以執行該匯入 (IMPORT) 程式以從檔案匯入金鑰至已建立的金鑰儲存體檔案。在執行程式前，您想匯入金鑰的金鑰儲存體檔案必須已經存在。

若要變更程式範例，請遵循下列步驟。

1. 將加密用之密碼鎖相同的未加密金鑰，匯入兩個產品中。對於「 CCA 服務」，加密用的密碼鎖金鑰必須爲匯出器 (EXPORTER)，而對於 CCA CSP 它必須爲匯入器 (IMPORTER)。
2. 爲您想移轉的**每一個金鑰**，於「CCA 服務」中執行 Key_Export (CSNBKEX) CCA API。這會使程式範例呼叫 API。
3. 藉由使用 Key_Import (CSNBKIM) CCA API，匯入輸出的外部金鑰記號至 CCA CSP 及「4758 輔助處理器」。請記得變更程式，來爲**每一個金鑰**執行此步驟。

一旦您變更程式以定址每一個金鑰，您就可執行該程式。請記得先執行匯出 (EXPORT) 然後執行匯入 (IMPORT)。

註: 如果您選擇使用提供的程式範例，請變更它們以滿足您特定的需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

範例：匯出 (EXPORT) 金鑰: 這是步驟一。變更這個程式範例以滿足您移轉金鑰儲存檔案的需要。一旦您執行此程式，請使用第 100 頁的『範例：匯入 (IMPORT) 金鑰』來完成移轉處理。

```
/*-------------------------------------------------------------*/
/* 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;
}
```

範例：匯入 (IMPORT) 金鑰: 此爲步驟二。如果您尚未這樣做，請執行第 95 頁的『範例：匯出 (EXPORT) 金鑰』程式以啓動移轉程序。然後變更此程式範例，以滿足您完成移轉金鑰儲存檔案的需要。

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

}
```

為 iSeries 跨網域金鑰檔案移轉「密碼支援」

若之前已於您的伺服器上使用加密，您可能已有來自 iSeries (5769-CR1) 的「密碼支援」之加密跨網域檔案。您可以將現有的跨網域金鑰移轉到新的「4758 輔助處理器」。

iSeries 產品 (5769-CR1 或 5722-CR1) 的「密碼支援」會在主要金鑰下加密其跨網域金鑰，並且將它們儲存到檔案中。「共同密碼架構 (CCA)」無法在此套表中使用它們，但是您可以從 CCA 的「密碼支援」產品中移轉它們，以與「輔助處理器」一起使用。在完成此作業之前，您必須考慮一些事情：

- **由跨網域金鑰加密的跨網域金鑰**。iSeries 的「密碼支援」支援匯入跨網域金鑰之清除的金鑰值及在跨網域金鑰下加密資料金鑰。但是，它不支援跨網域金鑰下加密跨網域金鑰，也不支援傳回任何跨網域金鑰的清除的金鑰值。由此，移轉跨網域金鑰是相當複雜的，而不只是執行匯出和匯入作業。

- **單倍金鑰對雙倍金鑰**。 iSeries 「密碼支援」中的所有金鑰爲單倍金鑰。於 CCA 中，所有加密用的金鑰及個人識別號碼 (PIN) 金鑰都是雙倍金鑰。儘管金鑰的長度不同，您仍可以由單倍金鑰建置雙倍金鑰，及使雙倍金鑰像單倍金鑰一樣運作。如果雙倍金鑰的兩半是相同的，則任何加密作業的結果將會相同，就好象使用單倍金鑰一樣。因此，當您由 iSeries 的「密碼支援」移轉金鑰給 CCA 時，會需要將跨網域金鑰的金鑰值複製到 CCA 金鑰之金鑰值的兩半中。
- **CCA 控制向量對主要金鑰變式**。於 CCA 中，當金鑰要在加密用的金鑰下加密時，它實際上是在由此加密用的金鑰和控制向量的互斥 OR 作業所形成的金鑰下加密。若爲「密碼支援」，跨網域金鑰在三個不同的主要金鑰變式之一下加密。主要金鑰變式是主要金鑰 (8 位元組的十六進位 22、44 或 88) 互斥 OR 作業的結果。控制向量和主要金鑰變式都提供了金鑰分離，從而限制金鑰的使用意圖。於 CCA 中，控制向量的值決定了它的使用。於「密碼支援」中，怎樣使用金鑰決定了將使用哪一個主要金鑰變式來解密它。於此兩種情況中，違背金鑰使用意圖的任何使用金鑰之嘗試都將導致錯誤。儘管控制向量和主要金鑰變式可能起類似的作用，但是用於形成主要金鑰變式的值與控制向量不同。
- **不對稱的雙倍金鑰之 CCA 控制向量**。只有在雙倍金鑰的兩半相同時，雙倍金鑰與單倍金鑰行爲才相同。雙倍金鑰的控制向量不對稱。任何雙倍金鑰與控制向量互斥 OR 都不會導致具有相同兩半的金鑰。此雙倍金鑰與單倍金鑰將有不同的行爲。

您可以選擇兩種方法之一來移轉金鑰。

方法 1 (建議)

此方法爲以上列出的注意事項提供了部份解決方案，其爲建議使用的方法。

若要將跨網域金鑰從「密碼支援」移轉到 CCA，您需要使用對兩者都通用的加密用的金鑰。可以使用「密碼支援」主要金鑰作爲「密碼支援」及 CCA 之間的共同金鑰 (在 CCA 中，host master key (主要金鑰) 就是所謂的 master key (主要金鑰))。將「密碼支援」主要金鑰清除值匯入 CCA 作爲匯入器 (IMPORTER) 加密用的金鑰。因爲您將主要金鑰輸入爲兩個分開的組件，所以應該考慮使用 Key_Part_Import (CSNBKPI) CCA API 將它作爲兩個組件匯入 CCA。若您對「密碼支援」主要金鑰有雙重責任，則應該維護此加密用的金鑰的雙重責任。另外，若您知道主要金鑰的兩個組件，則您也可以執行兩個組件的互斥 OR，並且將金鑰作爲一個組件匯入。該程式範例使用此方法匯入主要金鑰。您可能考慮將匯入主要金鑰作爲完全獨立的處理，而不是如程式範例那樣與移轉所有跨網域金鑰相聯繫。

有三種跨網域金鑰類型：

- 接收跨網域金鑰
- 傳送跨網域金鑰
- 個人識別號碼 (PIN) 跨網域金鑰

接收跨網域金鑰的 CCA 等價體爲匯入器 (IMPORTER) 加密用的金鑰。兩者皆用於接收或匯入加密金鑰。

傳送跨網域金鑰用於兩者：a) 加密資料金鑰，之後可被傳送至另一個系統；及 b) 轉換加密的個人識別碼 (PIN)。CCA 擁有比「密碼支援」產品更嚴格的金鑰隔離，因此無法產生或匯入提供全部兩個功能的金鑰。若該金鑰既用於匯出器 (EXPORTER) 加密用的金鑰也用於 OPINENC (離埠個人識別號碼 (PIN) 加密) 金鑰，則您需要兩次將傳送跨網域金鑰匯入到兩個具有不同金鑰類型的金鑰。

您可將個人識別號碼 (PIN) 跨網域金鑰用於產生個人識別號碼 (PIN) 和驗證個人識別號碼 (PIN)。 CCA 將這兩種用法區分爲 PINGEN (個人識別號碼 (PIN) 產生) 和 PINVER (個人識別號碼 (PIN) 驗證) 金鑰。若此金鑰同時用於產生和驗證個人識別號碼 (PIN)，則您同樣需要匯入個人識別號碼 (PIN) 跨網域金鑰兩次。

主要金鑰加密資料金鑰，而其它的主要金鑰變式加密跨網域金鑰。

- 主要金鑰變式 1 加密傳送跨網域金鑰。變式 1 爲主要金鑰與 8 位元組的十六進位數 88 之互斥 OR 作業的結果。
- 主要金鑰變式 2 加密接收跨網域金鑰。變式 2 爲主要金鑰與 8 位元組的十六進位數 22 之互斥 OR 作業的結果。
- 主要金鑰變式 3 加密個人識別號碼 (PIN) 跨網域金鑰。變式 3 爲主要金鑰與 8 位元組的十六進位數 44 之互斥 OR 作業的結果。

註：若只想將主要金鑰的未加密金鑰值匯入 CCA，則無法移轉任何金鑰。爲了移轉金鑰，您需要將加密它的主要金鑰變式考慮在內。

建立主要金鑰變式的 8 位元組值類似於控制向量。移轉金鑰的處理可視爲變更金鑰上的控制向量。 IBM 4758 CCA IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide 說明這樣的處理之方法。此方法爲預先互斥 OR 技術。若匯入金鑰前，加密用之金鑰 (此情況中爲主要金鑰) 的未加密金鑰值已與控制向量資訊進行互斥 OR，則您可以有效變更由此加密用的金鑰匯入的任何金鑰之控制向量。

若您使用單倍金鑰，則預先互斥 OR 技術將運行良好。對於雙倍金鑰，因爲 CCA 金鑰左右兩半的控制向量並不相同，所以必須變更此技術。若要克服此差異，如下匯入金鑰兩次：

1. 建立一個 16 位元組值，使每一半 (8 位元組) 與要匯入金鑰的控制向量之左半部份相同。在預先互斥 OR 技術中使用此 16 位元組值以建立匯入器加密用的金鑰，可稱之爲「左匯入器」。使用此加密用的金鑰匯入的金鑰僅左半部份是有效的。
2. 建立另一個 16 位元組值，使每一半 (8 位元組) 與要匯入金鑰的控制向量之右半部份相同。在預先互斥 OR 技術中使用此 16 位元組值以建立匯入器加密用的金鑰。使用此匯入器加密用的金鑰，已匯入的金鑰僅右半部份是有效的
3. 匯入跨網域兩次：
   a. 首先，使用步驟 1 中建立的加密用的金鑰，儲存結果的左半部份。
   b. 然後，使用步驟 2 中建立的加密用的金鑰，儲存結果的右半部份。
4. 最後，連接步驟 A 中結果的左半部份和步驟 B 中結果的右半部份。將合併的結果置於新的金鑰記號中。

您現在便有一個 CCA 雙倍金鑰，它類似於來自 iSeries「密碼支援」產品中的跨網域金鑰。

第 119 頁的『使用匯入器 (IMPORTER) 加密用的金鑰』彙總了匯入所有跨網域金鑰時所需要的所有匯入器之加密用的金鑰。它還說明了如何建立匯入器的加密用的金鑰。

方法 2

註: 您應該僅於系統和環境的安全性絕無問題的情況下使用此方法。此方法比建議方法更加簡單，但對於跨網域金鑰檔案它有較大的安全性風險，因爲跨網域金鑰將在應用程式儲存體中成爲清除格。

1. 藉由使用 Clear_Key_Import (CSNBCKI) CCA API 把主要金鑰作爲資料金鑰匯入 CCA。對於含有產生主要金鑰變式之等價資料金鑰所需的值之金鑰，請記得對其執行互斥 OR 運算，如下所示：
   - 主要金鑰變式 1 加密傳送跨網域金鑰。變式 1 爲主要金鑰與 8 位元組的十六進位數 88 之互斥 OR 作業的結果。
   - 主要金鑰變式 2 加密接收跨網域金鑰。變式 2 爲主要金鑰與 8 位元組的十六進位數 22 之互斥 OR 作業的結果。
   - 主要金鑰變式 3 加密個人識別號碼 (PIN) 跨網域金鑰。變式 3 爲主要金鑰與 8 位元組的十六進位數 44 之互斥 OR 作業的結果。

   此步驟後您將有三種不同的資料金鑰。
2. 使用 Decrypt (CSNBDEC) CCA API 來解密跨網域金鑰以傳回未加密金鑰值。使用正確資料金鑰以對其解密。
3. 使用 Key_Part_Import (CSNBKPI) CCA API 將未加密金鑰匯入 CCA。

您應該知道此方法並不安全。在此方法期間的一些時間中，所有金鑰都會位於應用程式儲存體中的清除套表中。

恭喜您！現在您已經可以勝任撰寫移轉跨網域金鑰的程式，或者變更以下程式範例。

範例：將跨網域金鑰移轉到「4758 輔助處理器」，以移轉加密支援跨網域金鑰： 請變更此程式範例，以滿足您將 iSeries 跨網域金鑰檔案的「密碼支援」移轉到「4758 輔助處理器」的需要。

```
/*********************************************************************/
/* 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;
}
```

使用匯入器 (IMPORTER) 加密用的金鑰: 若要匯入全部類型的跨網域金鑰，您將需要下列 IMPORTER 加密用的金鑰：

1. 匯入匯出器金鑰左半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是匯出器加密用的金鑰控制向量的左半部份，爲十六進位 88 的 16 位元組。

2. 匯入匯出器金鑰右半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是匯出器加密用的金鑰控制向量的右半部份，爲十六進位 88 的 16 位元組。

3. 匯入匯入器金鑰左半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是匯入器加密用的金鑰控制向量的左半部份，爲十六進位 22 的 16 位元組。

4. 匯入匯入器金鑰右半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是匯入器加密用的金鑰控制向量的右半部份，爲十六進位 22 的 16 位元組。

5. 匯入 OPINENC 金鑰左半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是 OPINENC 金鑰控制向量的左半部份，爲十六進位 88 的 16 位元組。

6. 匯入 OPINENC 金鑰右半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是 OPINENC 金鑰控制向量的右半部份，爲十六進位 88 的 16 位元組。

7. 匯入 IPINENC 金鑰左半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是 IPINENC 金鑰控制向量的左半部份，爲十六進位 44 的 16 位元組。

8. 匯入 IPINENC 金鑰右半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是 IPINENC 金鑰控制向量的右半部份，爲十六進位 44 的 16 位元組。

9. 匯入 PINGEN 金鑰左半部份的 KEK。

   使用清除的主要金鑰建立此金鑰，它是 PINGEN 金鑰控制向量的左半部份，爲十六進位 44 的 16 位元組。

10. 匯入 PINGEN 金鑰右半部份的 KEK。

    使用清除的主要金鑰建立此金鑰，它是 PINGEN 金鑰控制向量的左半部份，爲十六進位 44 的 16 位元組。

11. 匯入 PINVER 金鑰左半部份的 KEK。

    使用清除的主要金鑰建立此金鑰，它是 PINVER 金鑰控制向量的左半部份，爲十六進位 44 的 16 位元組。

12. 匯入 PINVER 金鑰右半部份的 KEK。

    使用清除的主要金鑰建立此金鑰，它是 PINVER 金鑰控制向量的左半部份，爲十六進位 44 的 16 位元組。

移轉金鑰儲存檔案

此處的程序..解譯由 iSeries 服務的「共同密碼架構密碼服務提供程式 CCCA (SP)」移轉金鑰儲存檔案的處理程序。

管理 4758 加密輔助處理器

此節主要是有關「4758 輔助處理器」的 OS/400 應用程式使用。如果您正在使用具有 SSL 的多個「輔助處理器」，請參閱第 176 頁的『管理多重 4758 加密輔助處理器』及第 186 頁的『複製主要金鑰』。

在您設定「4758 輔助處理器」後，可以開始撰寫程式以使用「4758 輔助處理器」的加密功能。您可使用程式以執行這些作業：

- 『登入或登出 4758 加密輔助處理器』使用受角色限制的 API。
- 第 131 頁的『查詢狀態或要求資訊』.
- 第 136 頁的『起始設定金鑰儲存檔案』如果您計畫保存 DES 和 PKA 金鑰的記錄。
- 第 141 頁的『建立 DES 與 PKA 金鑰』並且將它們儲存在 DES 金鑰儲存中。
- 第 148 頁的『加密或解密檔案』.
- 第 154 頁的『使用 PIN』.
- 第 167 頁的『產生並驗證數位簽章』.
- 第 176 頁的『管理多重 4758 加密輔助處理器』.
- 第 186 頁的『複製主要金鑰』當使用多個「4758 輔助處理器」時。

註: 此節中的許多頁都包括一個或多個程式範例。變更這些程式以滿足您的特定需要。部份程式需要僅變更一個或兩個參數，而其餘程式需要則要進一步的變更。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

登入或登出 4758 加密輔助處理器

登入

僅當您想使用採用預設角色中未啓用的存取控制點的 API 時，才需要登入。使用適當角色的設定檔登入，該角色啓用了您要使用的存取控制點。

登入「4758 輔助處理器」後，就可執行程式以利用「4758 輔助處理器」的加密功能。您可藉由撰寫使用 Logon_Control (CSUALCT) API 動詞的應用程式來登入。提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 『範例：用於登入 4758 輔助處理器的 ILE C 程式』
- 第 123 頁的『範例：用於登入 4758 輔助處理器的 ILE RPG 程式』

登出

當您完成使用「4758 輔助處理器」後，應該登出「4758 輔助處理器」。您可以藉由撰寫使用 Logon_Control (CSUALCT) API 動詞的應用程式來登出。提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 第 126 頁的『範例：登出 4758 輔助處理器的 ILE C 程式』
- 第 128 頁的『範例：登出 4758 輔助處理器的 ILE RPG 程式』

註: 如果您選擇使用提供的程式範例，請變更它們以滿足您特定的需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

範例：用於登入 4758 輔助處理器的 ILE C 程式

變更此程式範例，以滿足您登入「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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");
}
```

範例：用於登入 4758 輔助處理器的 ILE RPG 程式

變更此程式範例，以滿足您登入「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

範例：登出 4758 輔助處理器的 ILE C 程式

請變更此程式範例，以滿足您登出「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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,
         verb_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);
   }
}
```

範例：登出 4758 輔助處理器的 ILE RPG 程式

請變更此程式範例，以滿足您登出「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

查詢狀態或要求資訊

您可以查詢「4758 輔助處理器」以決定性質，例如已啓用哪些演算法，它支援的金鑰長度、主要金鑰狀態、複製狀態及時鐘設定。查詢「4758 輔助處理器」的最容易且最快速的方法是使用「4758 加密輔助處理器」配置 Web 型公用程式。按一下**顯示配置**，然後選取裝置，再選取您要顯示的項目。

如果您偏好撰寫自己的應用程式來查詢「輔助處理器」，可以藉由使用 Cryptographic_Facility_Query (CSUACFQ) API 動詞來執行之。提供兩個範例程式，供您參考。『範例：查詢 4758 輔助處理器的狀態』使用 STATEID 及 TIMEDATE 關鍵字，同時第 134 頁的『範例：要求來自 4758 輔助處理器的資訊』向使用者提示第二個必要的關鍵字。

IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide 說明 Cryptographic_Facility_Query (CSUACFQ) 安全性應用程式設計介面、您可以要求的資訊類型，及所傳回資訊的格式。

範例：查詢 4758 輔助處理器的狀態

變更此程式範例，以滿足查詢「4758 輔助處理器」狀態的需要。

```
/*-------------------------------------------------------------------*/
/* 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);
    }
}
```

範例：要求來自 4758 輔助處理器的資訊

變更此程式範例，以滿足要求自「4758 輔助處理器」的資訊之需要。

```
/*-------------------------------------------------------------------*/
/* 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);
     }

}
```

起始設定金鑰儲存檔案

金鑰儲存檔案爲儲存作業金鑰的資料庫檔案，亦即，主要金鑰下加密的金鑰。您可以起始設定兩種不同類型的「4758 輔助處理器」之金鑰儲存。「4758 輔助處理器」使用一種類型來儲存 PKA 金鑰，使用另一種類型來儲存 DES 金鑰。如果您計畫將金鑰儲存在金鑰儲存檔案中，或者您計劃在硬體上使用保留金鑰，則需要起始設定金鑰儲存檔。

如果金鑰儲存檔案尚未存在，則 CCA CSP 會建立一個 DB2® 金鑰儲存檔案。如果金鑰儲存檔案已存在，則 CCA CSP 會刪除此檔案並重建一個新的金鑰儲存檔案。

若要起始設定金鑰儲存，可使用「4758 加密輔助處理器」配置公用程式。在**管理配置**上按一下，然後在 **DES 金鑰**或 **PKA 金鑰**上按一下，這視您想要起始設定的金鑰儲存檔案而定。使用公用程式，您僅可以起始設定一個檔案 (若此檔案尚未存在)。

如果您寧願撰寫自己的應用程式來起始設定金鑰儲存檔案，則可使用 KeyStore_Initialize (CSNBKSI) API 動詞來這樣做。提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 第 137 頁的『起始設定範例：「4758 輔助處理器」的金鑰儲存之 ILE C 程式』
- 第 139 頁的『範例：起始設定「4758 輔助處理器」的金鑰儲存之 ILE RPG 程式』

註: 如果您選擇使用提供的程式範例之一，則請變更它以滿足您的特定需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

在建立「4758 輔助處理器」的金鑰儲存後，您可以使用第 141 頁的『建立 DES 與 PKA 金鑰』產生 DES 與 PKA 金鑰以儲存在金鑰儲存檔案中。

起始設定範例：「4758 輔助處理器」的金鑰儲存之 ILE C 程式

變更此程式範例，以滿足起始設定「4758 輔助處理器」的金鑰儲存之需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-----------------------------------------------------------------*/
/* 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;
  }
 }
```

範例：起始設定「4758 輔助處理器」的金鑰儲存之 ILE RPG 程式

變更此程式範例，以滿足起始設定「4758 輔助處理器」的金鑰儲存之需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

建立 DES 與 PKA 金鑰

您可以使用「4758 輔助處理器」來建立兩種類型的加密金鑰。

- 「資料加密標準 (DES)」金鑰的內容基於對稱演算法。這表示加密使用相同的金鑰值來加密及解密資料。爲第 148 頁的『加密或解密檔案』、第 154 頁的『使用 PIN』和管理金鑰使用 DES 金鑰。

  若要用「4758 輔助處理器」建立 DES 金鑰，請撰寫程式或變更此程式第 142 頁的『範例：用 4758 輔助處理器建立 DES 金鑰』。

- 公開金鑰演算法 (PKA) 金鑰的內容基於不對稱演算法，這表示加密使用不同的金鑰來加密及解密。爲使用第 167 頁的『產生並驗證數位簽章』的簽署檔案和管理金鑰使用 PKA 金鑰。

  若要用「4758 輔助處理器」建立 PKA 金鑰，請撰寫程式或變更此第 145 頁的『範例：以 4758 輔助處理器建立 PKA 金鑰』。

註: 如果您選擇使用提供的程式範例，請變更它們以滿足您特定的需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

使用第 136 頁的『起始設定金鑰儲存檔案』，將 DES 和 PKA 金鑰儲存到爲它們建立的金鑰儲存檔案中。您也可以將 PKA 金鑰儲存到「4758 輔助處理器」中。請參閱 http://www.ibm.com/security/cryptocards/html/library.shtml 上的 4758 資訊，以取得在硬體中儲存金鑰的相關資訊。

範例：用 4758 輔助處理器建立 DES 金鑰

變更此程式範例，使之滿足以「4758 輔助處理器」建立 DES 金鑰的需要。

```
/*-------------------------------------------------------------*/
/* 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;
  }
}
```

範例：以 4758 輔助處理器建立 PKA 金鑰

變更此程式範例，使之滿足以「4758 輔助處理器」建立 PKA 金鑰的需要。

```
/*-------------------------------------------------------------*/
/* 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;
  }
}
```

加密或解密檔案

「4758 輔助處理器」的其中一個較實用的用途爲加密及解密資料檔。您可以使用這些加密方法的其中一個來保護檔案：

- 將整個檔案作爲位元組字串處理 (這是程式範例使用的方法)。
- 加密每一個記錄或每一個記錄的每一個部份。

在此程式『範例：以 4758 輔助處理器加密資料』中撰寫您自己的程式或變更技術，來保護許多不同格式的資料，不僅是資料檔。

範例：以 4758 輔助處理器加密資料

變更此程式範例，使之滿足以「4758 輔助處理器」加密資料的需要。

```
/*-------------------------------------------------------------------*/
/*                                                                   */
/* 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
                                               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
                                                    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;
}
```

使用 PIN

金融機構使用個人識別碼 (PIN) 來爲其客戶授權個人金融交易。PIN 與密碼類似，除了它包含十進位數且通常是相關帳戶號碼的加密函數之外。您可以使用「4758 輔助處理器」來使用 PIN。

若要使用 PIN，請撰寫一個程式或變更此『範例：於 4758 輔助處理器上使用 PIN』。

註: 如果您選擇使用提供的程式範例，請變更它以滿足特定的需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

範例：於 4758 輔助處理器上使用 PIN

變更此程式範例，以滿足於「4758 輔助處理器」上使用 PIN 的需要。

```
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'     FAILMSG
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'     FAILMSG
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
```

產生並驗證數位簽章

產生數位簽章

藉由併入稱爲數位簽章的識別值證明，可以保護資料免受未能偵測的變更。數位簽章依賴於雜湊法與公開金鑰加密法。當您簽署資料時，將雜湊資料並用您的私密金鑰對結果加密。加密的雜湊值稱爲數位簽章。

如果變更原始資料，則會產生一個不同的數位簽章。

若要使用 PKA 金鑰來簽署檔案，請撰寫程式或變更此程式『範例：使用 4758 輔助處理器來簽署檔案』。

驗證數位簽章

驗證數位簽章與簽署資料相反。驗證簽章會告知您是否變更了已簽署的資料。驗證數位簽章時，會使用公開金鑰解密該簽章以產生原始雜湊值。已簽署的資料是雜湊的。如果這兩個雜湊值相符，則該簽章已被驗證。若要進行驗證，請撰寫程式或變更此程式第 171 頁的『範例：使用 4758 輔助處理器來驗證數位簽章』。

註: 如果您選擇使用提供的程式範例，請變更它們以滿足您特定的需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

範例：使用 4758 輔助處理器來簽署檔案

請變更此程式範例，以滿足您以「4758 輔助處理器」來簽署檔案的需要。

```
/*-------------------------------------------------------------*/
/* 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;
   }
}
```

範例：使用 4758 輔助處理器來驗證數位簽章

請變更此程式範例，以滿足您以「4758 輔助處理器」來驗證數位簽章的需要。

```
/*--------------------------------------------------------------*/
/* 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;
    }
}
```

管理多重 4758 加密輔助處理器

在每一個系統上您最多可以有 8 個「4758 輔助處理器」。如果多重「4758 輔助處理器」全部配置爲相同的話，則將工作分佈到多重「4758 輔助處理器」和多重工作中會提供您更好的效能。在某個時間中僅可以將一個「輔助處理器」(密碼裝置說明) 配置給一個工作。但是，可以藉由取消配置現行「輔助處理器」並配置一個新的來在「輔助處理器」之間切換工作。對於 OS/400 SSL 使用者，如果 DCM 中的 SSL 配置指出多於一個的「輔助處理器」將用於 SSL 階段作業建立，則「輔助處理器」的配置與取消配置會由系統來管理。

如果將所有的「輔助處理器」配置爲相同，則所有作業金鑰將於所有「輔助處理器」上執行同樣的工作。一個「輔助處理器」上的任何加密資料可以在不同的「輔助處理器」上解密。所有金鑰儲存檔案在任何「輔助處理器」上都是可交換的。將「輔助處理器」配置爲相同的最重要部份爲主要金鑰。如果您在一個「輔助處理器」的組件中輸入主要金鑰，並且希望它們能夠可交換地工作，則您必須在所有其它「輔助處理器」的組件中輸入相同的主要金鑰。如果在「輔助處理器」內部產生隨機主要金鑰，並且您希望所有「輔助處理器」能夠可交換地工作，則您必須將該主要金鑰複製到其它「輔助處理器」。

在某些特定狀況中，您可能不希望所有「輔助處理器」配置爲相同。它們可以全部有不同的配置，或可以設立爲群組，群組內部配置相同，而群組之間則不同。對於這些案例，所有作業金鑰可能不會在所有「輔助處理器」上相同地工作。一個「輔助處理器」上的加密資料可能無法在不同的「輔助處理器」上回復。此外，金鑰庫檔案可能無法在「輔助處理器」之間互換地工作。對於這些狀況，您必須記錄哪些金鑰庫檔案及作業金鑰可作用於給定「輔助處理器」的追蹤。雖然將「輔助處理器」配置爲不同可能限制加密應用程式的可擴展性，但是它可以在安全性方面提供更小的粒度。例如，您可以授予不同的物件權限給不同的密碼裝置說明。

如果您使用保留的 PKA 金鑰，則「輔助處理器」也是不可互換的。在「輔助處理器」的外部，無法以任何方式匯出保留金鑰。因此，使用保留金鑰的任何加密要求必須傳送至儲存保留金鑰的「輔助處理器」。

下列資源僅在您使用 OS/400 應用程式時是適用的：

配置裝置

Cryptographic_Resource_Allocate (CSUACRA) API 動詞用於明確地將加密裝置配置給工作，以便系統可以決定如何遞送所有後續的加密要求。如果您在使用任一 CCA API 動詞之前，沒有先明確地使用 Cryptographic_Resource_Allocate (CSUACRA) API 動詞，則系統將嘗試配置預設密碼裝置。預設裝置是名爲 CRP01 的密碼裝置。必須藉由使用「基本」配置精靈或「建立裝置密碼 (CRTDEVCRP)」CL 指令來建立它。當您希望使用不是預設密碼裝置的裝置時，才需要使用 CSUACRA。不論是明確地還是隱含地配置工作的裝置，都會保持配置直到工作結束，或者直到使用 Cryptographic_Resource_Deallocate (CSUACRD) API 動詞解除配置該裝置爲止。提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩個程式皆執行相同的功能。

- 『範例：配置輔助處理器的 ILE C 程式』
- 第 179 頁的『範例：配置輔助處理器的 ILE RPG 程式』

取消配置裝置

當您結束使用「4758 輔助處理器」時，您應該藉由使用 Cryptographic_Resource_Deallocate (CSUACRD) API 動詞來取消配置「4758 輔助處理器」。除非使用密碼裝置的所有工作都已經解除配置該密碼裝置，否則無法轉斷該密碼裝置說明。提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩個程式皆執行相同的功能。

- 第 181 頁的『範例：解除配置輔助處理器的 ILE C 程式』
- 第 183 頁的『範例：解除配置輔助處理器的 ILE RPG 程式』

範例：配置輔助處理器的 ILE C 程式

請變更此程式範例，以滿足您配置「輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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);
    }
}
```

範例：配置輔助處理器的 ILE RPG 程式

請變更此程式範例，以滿足您配置「輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
     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
```

範例：解除配置輔助處理器的 ILE C 程式

變更此程式範例，以滿足您解除配置「輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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);
    }
}
```

範例：解除配置輔助處理器的 ILE RPG 程式

變更此程式範例，以滿足您解除配置「輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
      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
```

複製主要金鑰

主要金鑰複製是將主要金鑰從一個「4758 輔助處理器」安全地複製到另一個「4758 輔助處理器」，且不外曝主要金鑰值的方法。可透過將主要金鑰分割爲 *n* 個共用的處理來執行此方法，其中 *n* 是介於 1 至 15 間的數字。重新建置另一個「輔助處理器」中的主要金鑰時需要 *m* 個共用，其中 *m* 是介於 1 至 15 間且小於或等於 *n* 的數字。

術語「複製 (cloning)」用於區分「複製 (copying)」處理，因爲沒有一個共用或少於 *m* 個共用的任意組合可提供重新建置主要金鑰所需的足夠資訊。

包含將被複製之主要金鑰的「輔助處理器」被稱爲主要金鑰共用來源節點或「傳送器」。「傳送器」必須產生保留的 RSA 金鑰對。當產生私密金鑰時，必須將此私密金鑰標示爲適合與複製一起使用。金鑰被稱爲「輔助處理器共用簽署」金鑰或「傳送器」金鑰。要接收主要金鑰的「輔助處理器」被稱爲主要金鑰共用目標節點或「接收器」。「接收器」也必須產生保留的 RSA 金鑰對，且必須將其標示爲適於與複製一起使用。此金鑰被稱爲「輔助處理器共用接收」金鑰或僅是「接收器」金鑰。

「傳送器」與「接收器」公開金鑰兩者必須由「輔助處理器」(被稱爲公開金鑰認證節點或「認證者」)中的保留私密金鑰來數位簽署或認證。此保留私密金鑰爲「驗證者」金鑰。它亦被稱爲「共用管理」金鑰。在產生並接收共用之前必須在「傳送器」及「接收器」中登記相關的公開金鑰。 「4758 輔助處理器」可能僅具有「驗證者」的角色，或者既是「驗證者」又是「傳送器」，或者既是「驗證者」又是「接收器」。

當產生每一個共用時，該共用由「輔助處理器」使用「傳送器」私密金鑰進行簽署，並由最近產生的三重 DES 演算法金鑰進行加密。然後「接收器」公開金鑰覆蓋或加密三重 DES 演算法金鑰。

當接收每一個共用時，使用「傳送器」公開金鑰驗證共用上的簽章，使用「接收器」私密金鑰解開或解密三重 DES 演算法金鑰，且使用三重 DES 演算法金鑰解密共用。當已接收到 m 個共用時，將在「接收器」的新主要金鑰註冊中完成已複製的主要金鑰。

複製主要金鑰最容易且最快的方法是使用「4758 加密輔助處理器」配置 Web 型公用程式。公用程式包括「主要」金鑰複製通告器。若要啓動主要金鑰複製通告器，請遵循下列步驟：

1. 在「4578 加密輔助處理器」配置頁上按一下**管理配置**。
2. 按一下**主要金鑰**。
3. 選取一個裝置。
4. 輸入有效的「輔助處理器」設定檔及密碼。
5. 按一下**複製**按鈕。

如果您偏好撰寫自己的應用程式來複製主要金鑰，您可以使用下列 API 動詞：

- Cryptographic_Facility_Control (CSUACFC)
- PKA_Key_Token_Build (CSNDPKB) (可能不需要，取決於您如何撰寫應用程式)
- PKA_Key_Generate (CSNDPKG)
- PKA_Public_Key_Register (CSNDPKR)
- One_Way_Hash (CSNBOWH)
- Digital_Signature_Generate (CSNDDSG)
- Master_Key_Distribution (CSUAMKD)

提供了九對範例程式，以供您參考。每一對包含以 ILE C 撰寫的程式及以 ILE RPG 撰寫的程式。兩者皆執行相同的功能。

- 『範例：於 4758 輔助處理器中設定主要金鑰共用的最小值及最大值的 ILE C 程式』
- 第 189 頁的『範例：於 4758 輔助處理器中設定主要金鑰共用之最小與最大值的 ILE RPG 程式』
- 第 192 頁的『範例：爲複製主要金鑰產生一個保留的金鑰對之 ILE C 程式』
- 第 197 頁的『範例：爲複製主要金鑰而產生保留金鑰對的 ILE RPG 程式』
- 第 204 頁的『範例：註冊公開金鑰雜湊的 ILE C 程式』
- 第 207 頁的『範例：註冊公開金鑰雜湊的 ILE RPG 程式』
- 第 213 頁的『範例：註冊公開金鑰憑證的 ILE C 程式』
- 第 216 頁的『範例：註冊公開金鑰憑證的 ILE RPG 程式』
- 第 220 頁的『範例：認證公開金鑰記號的 ILE C 程式』
- 第 225 頁的『範例：認證公開金鑰記號的 ILE RPG 程式』
- 第 233 頁的『範例：獲得主要金鑰共用的 ILE C 程式』
- 第 236 頁的『範例：獲得主要金鑰共用的 ILE RPG 程式』
- 第 242 頁的『範例：安裝主要金鑰共用的 ILE C 程式』
- 第 246 頁的『範例：安裝主要金鑰共用的 ILE RPG 程式』

剩餘的兩對範例程式對於主要金鑰複製不是必需的。但它們對於開發及測試先前的範例程式可能有用。

- 第 253 頁的『範例：列出保留金鑰的 ILE C 程式』
- 第 255 頁的『範例：列出保留金鑰的 ILE RPG 程式』
- 第 258 頁的『範例：刪除保留金鑰的 ILE C 程式』
- 第 260 頁的『範例：刪除保留金鑰的 ILE RPG 程式』

如需複製主要金鑰的相關資訊，請參照 IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide。

範例：於 4758 輔助處理器中設定主要金鑰共用的最小值及最大值的 ILE C 程式

請變更此程式範例，以滿足您在「 4758 輔助處理器」中設定主要金鑰共用的最小值及最大值的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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);
    }
}
```

範例：於 4758 輔助處理器中設定主要金鑰共用之最小與最大值的 ILE RPG 程式

請變更此程式範例，以滿足您在「 4758 輔助處理器」中設定主要金鑰共用的最小值及最大值的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

範例：為複製主要金鑰產生一個保留的金鑰對之 ILE C 程式

變更此程式範例，以滿足爲複製主要金鑰而產生保留金鑰對的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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;
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;
}
```

範例：為複製主要金鑰而產生保留金鑰對的 ILE RPG 程式

變更此程式範例，以滿足爲複製主要金鑰而產生保留金鑰對的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

範例：註冊公開金鑰雜湊的 ILE C 程式

變更此程式範例，以滿足註冊公開金鑰憑證雜湊的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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);

}
```

範例：註冊公開金鑰雜湊的 ILE RPG 程式

變更此程式範例，以滿足註冊公開金鑰憑證雜湊的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

範例：註冊公開金鑰憑證的 ILE C 程式

變更此程式範例，以滿足註冊公開金鑰憑證的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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);

}
```

範例：註冊公開金鑰憑證的 ILE RPG 程式

變更此程式範例，以滿足註冊公開金鑰憑證的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

範例：認證公開金鑰記號的 ILE C 程式

變更此程式範例，以滿足您認證公開金鑰記號的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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);
     }

}
```

範例：認證公開金鑰記號的 ILE RPG 程式

變更此程式範例，以滿足您認證公開金鑰記號的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

範例：獲得主要金鑰共用的 ILE C 程式

變更此程式範例，以滿足您獲得主要金鑰共用的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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;
  }

}
```

範例：獲得主要金鑰共用的 ILE RPG 程式

變更此程式範例，以滿足您獲得主要金鑰共用的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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
```

範例：安裝主要金鑰共用的 ILE C 程式

變更此程式範例，以滿足安裝主要金鑰共用的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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;
 }

}
```

範例：安裝主要金鑰共用的 ILE RPG 程式

變更此程式範例，以滿足安裝主要金鑰共用的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

範例：列出保留金鑰的 ILE C 程式

請變更此程式範例，以滿足您列出保留金鑰的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
/*-------------------------------------------------------------------*/
/* 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]);
       }
     }
   }
}
```

範例：列出保留金鑰的 ILE RPG 程式

請變更此程式範例，以滿足您列出保留金鑰的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
     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
[                                                                  ]
```

範例：刪除保留金鑰的 ILE C 程式

請變更此程式範例，以滿足您刪除保留金鑰的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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;
 }


}
```

範例：刪除保留金鑰的 ILE RPG 程式

請變更此程式範例，以滿足您刪除保留金鑰的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要的合法資訊。

```
     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 加密輔助處理器疑難排解

使用下列方法來處理使用「4758 輔助處理器」時可能發生的某些基本問題。如果疑難排解資訊不包含您的問題，請聯絡客戶服務代表。

始終確保您已爲相關產品及程式套用了所有現行的 PTF。

使用回覆碼

偵測及疑難排解問題的主要方法爲監督回覆碼及原因碼。

- **回覆碼 0** 指出已順利完成。爲了提供部份附加資訊，「4758 輔助處理器」將部份非零原因碼與此回覆碼相結合。
- **回覆碼 4** 指出應用程式設計介面 (API) 已完成處理，但有一個異常事件發生。它可能與應用程式所產生的問題相關，或者可能是基於提供給 API 的資料之正常出現。
- **回覆碼 8** 指出 API 未能順利完成。應用程式設計錯誤最有可能導致此狀況。
- **回覆碼 12** 通常指出在 4758 設定或配置中的部份問題類型。此碼表示 API 處理程序未順利完成。
- **回覆碼 16** 通常指出在「共同密碼架構密碼服務提供程式 (CCA CSP)」、iSeries 授權內碼或「4758 輔助處理器」授權內碼中的一個嚴重錯誤。對於這些錯誤類型，您應該聯絡您的客戶服務代表。

您也可以透過分析出現在工作日誌或系統操作員 (QSYSOPR) 佇列中的訊息，對問題進行疑難排解。通常，任何傳送訊息至工作日誌的事件，也會傳回一個相關的回覆碼及原因碼到呼叫程式。傳送給系統操作員佇列的訊息如果報告嚴重問題，則通常會指向有關此問題之附加資訊的來源。這樣的資訊是打算提供給 IBM 服務的，因此您可能不一定會發現它們對判斷問題是有用的。

一般錯誤

您應該提防這些一般錯誤：

- **您是否已安裝「4758 輔助處理器」及 CCA CSP**？ 「2620 加密處理器」和「2628 商用加密處理器」是使用 IBM CCA Services for iSeries 的完全獨立解決方案。「4758 輔助處理器」和 CCA CSP 將不使用 2620、2628 或 IBM CCA 服務。
- **您是否轉接裝置**？ 在轉接裝置之前，您都無法將任何要求傳送至「4758 輔助處理器」。
- **密碼裝置說明的資源是 「4758 加密輔助處理器」，還是 2620 或 2628**？ 它應該是「4758 加密輔助處理器」。
- **4758 輔助處理器是否找到裝置**？ 如果您尚未明確地使用 Cryptographic_Resource_Allocate API，則必須將密碼裝置命名爲 CRP01。如果未如此命名，則 CCA 無法選取任何裝置。命名裝置 CRP01 或變更程式爲使用 Cryptographic_Resource_Allocate CCA API，以選取裝置。
- **您是否選取正確的裝置**？ 如果您有一個預設裝置 (例如，名爲 CRP01 的裝置) 及一個附加裝置，則「4758 輔助處理器」將選取預設裝置，除非您使用 Cryptographic_Resource_Allocate。

- **「4758 輔助處理器」是否找到金鑰儲存檔案**？ 如果您尚未明確地使用 Key_Store_Designate SAPI，則 CCA CSP 支援將嘗試使用在裝置說明上命名的檔案。如果您在裝置說明上未命名任何檔案，則「4758 輔助處理器」將找不到任何檔案。
- **您是否已載入並設定主要金鑰**？ 除了用於配置「4758 輔助處理器」的加密要求之外，「4758 輔助處理器」將不會完成其它任何加密要求，除非您載入一個主要金鑰。
- **舊的主要金鑰註冊是否包含金鑰**？ 「4758 輔助處理器」不能在「現行的」主要金鑰下重新加密金鑰，除非「舊的」主要金鑰註冊包含值。
- **預設角色是否有權限使用給定的硬體指令**？ 如果沒有，則您將需要透過使用設定檔來登入，該設定檔使用有正確權限的角色。
- **角色是否有權限使用給定的硬體指令**？ 如果「4758 輔助處理器」需要硬體指令，但是您尚未授權角色來使用該指令，則您必須重新起始設定「4758 輔助處理器」。透過使用 Cryptographic_Facility_Control API 或在「系統服務工具」中找到的「硬體服務管理程式」來執行此項工作。使用 Cryptographic_Facility_Control API 需要您將重新起始設定「4758 輔助處理器」的硬體指令授權給一個角色。如果這樣的角色不存在，則您必須使用「硬體服務管理程式」。
- **是否已載入函數控制向量**？ 除了配置之外，「4758 輔助處理器」無法執行任何加密作業，直到您載入函數控制向量爲止。
- **是否已安裝密碼存取提供者產品之一**？ IBM 將函數控制向量與這些產品一同出貨。
- **如果您正在載入主要金鑰，則是否透過清除新的主要金鑰註冊來開始**？ 如果「4758 輔助處理器」已部份地載入新的主要金鑰註冊，則您無法載入主要金鑰的第一部份。
- **從 DEFAULT 角色移除權限之前，是否記得在 4758 中設定時鐘**？ 如果未設定時鐘，則您必須使用 Cryptographic_Facility_Control API 或使用於「系統服務工具」中找到的「硬體服務管理程式」來重新起始設定「4758 輔助處理器」。使用 Cryptographic_Facility_Control API 需要您將重新起始設定「4758 輔助處理器」的硬體指令授權給一個角色。如果這樣的角色不存在，則您必須使用「硬體服務管理程式」。
- **嘗試產生公用-專有金鑰對之前，您是否設定了 EID**？ 在可以產生 RSA 金鑰之前，您必須先設定 EID。
- **您是否正確地將空值金鑰記號的第一個位元組起始設定為二進位 0**？如果沒有，則 CCA 支援可能會嘗試將它作爲金鑰標籤來使用。 「CCA 支援」會將它作爲損壞的標籤格式來報告，或者報告可以找到金鑰記錄。
- **對於 PKA 金鑰儲存檔案和保留 PKA 金鑰中的標籤，您是否使用了相同的名稱**？ 如果使用了相同的名稱，則「4758 輔助處理器」將永遠找不到保留金鑰，因爲「4758 輔助處理器」總是最先搜尋金鑰儲存檔案。
- **在框架 PKA 金鑰記號中的任意欄位內是否有 EBCDIC 資料**？ 「4758 輔助處理器」特別地檢查一些欄位中的 ASCII 資料，且如果找到 EBCDIC 資料，則傳回錯誤。

如需進一步疑難排解資訊，請參閱『重新起始設定 4758 加密輔助處理器』和第 271 頁的『使用硬體服務管理程式』。

重新起始設定 4758 加密輔助處理器

如果您未正確地設定了「4758 輔助處理器」，則可能會以無法使用的配置結束，使用該配置您將無法執行任何加密功能並且無法使用任何 API 來回復。例如，您可以配置它，這樣您就沒有已授權設定主要金鑰的角色，也沒有已授權變更或建立新角色或設定檔的角色。

您可以透過使用 Cryptographic_Facility_Control (CSUACFC) SAPI，爲重新起始設定此卡呼叫硬體指令。提供兩個範例程式，供您參考。其中的一個以 ILE C 撰寫，而另一個以 ILE RPG 撰寫。兩者皆執行相同的功能。

- 『範例：重新起始設定 4758 輔助處理器的 ILE C 程式』
- 第 268 頁的『範例：重新起始設定 4758 輔助處理器的 ILE RPG 程式』

註: 如果您選擇使用所提供的程式範例，請變更它以滿足您特定的需要。因爲安全性理由，IBM 建議您爲這些程式範例個性化賦值而不是使用所提供的預設值。

但是，在一些情況下，可能沒有已授權可以使用任何硬體指令的角色。在這種情況下，您必須藉由使用「系統服務工具」中「硬體服務管理程式」所提供的功能，重新載入「授權內碼」，如第 271 頁的『使用硬體服務管理程式』中所說明。

更新「4758 輔助處理器」中的「授權內碼」

載入「4758 輔助處理器」中的「授權內碼」會消除儲存「4758 輔助處理器」中的主要金鑰、全部專用金鑰及全部角色與設定檔。因爲這樣，所以伺服器不會自動載入「4758 輔助處理器」中的「授權內碼」之 PTF，且 PTF 總是需要對組件採取動作以啓用它們。於載入「授權內碼」前，採用適當的動作以確保可以回復，例如確定您有主要金鑰的硬本。

註: 如果您隨機產生主要金鑰，則您會需要將該金鑰複製到另一個「4758 輔助處理器」中。如果不這麼做，則當您重新起始設定「4758 輔助處理器」時，將遺失全部加密金鑰。

範例：重新起始設定 4758 輔助處理器的 ILE C 程式

變更此程式範例，以滿足重新起始設定「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
/*--------------------------------------------------------------------*/
/* 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);
    }

}
```

範例：重新起始設定 4758 輔助處理器的 ILE RPG 程式

變更此程式範例，以滿足重新起始設定「4758 輔助處理器」的需要。

註: 請讀取第 281 頁的第 6 章, 『程式碼不保事項聲明』，以取得重要合法資訊。

```
     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.
```

使用硬體服務管理程式

硬體服務管理程式爲一個用來從邏輯與包裝角度來顯示及使用系統硬體的工具，一個用來除錯「輸入/輸出(I/O)」的處理器及裝置的輔助，且還可用來重新起始設定「4758 輔助處理器」(將它設定回未起始設定狀態)。

當重新起始設定「4758 輔助處理器」時，「4758 輔助處理器授權內碼」將被重新載入到「輔助處理器」中。部份而非所有的「輔助處理器」授權內碼的暫時修訂程式 (PTF) 可能需要使用硬體服務管理程式來啓動它們。併入此額外的步驟可讓您準備回復，因爲重新載入授權內碼的某些區段將導致遺失包括主要金鑰、保留的 RSA 私密金鑰、角色以及設定檔在內的任何配置資料。

可能會出現「4758 輔助處理器」必須重設回未初始化狀態的狀況。例如，如果未正確配置「輔助處理器」，則可能會出現「輔助處理器」無法執行任何有用功能，且無法使用「4758 輔助處理器」配置公用程式或使用者撰寫的應用程式來更正的實務。另一個範例是如果忘記了管理設定檔的密碼，且其他設定檔未使用已授權來變更密碼的角色。

在「系統服務工具」中可找到硬體服務管理程式。透過在 CL 指令行鍵入 STRSST 並按 Enter 鍵，來使用「啓動系統服務工具 (STRSST)」CL 指令。應顯示「系統服務工具登入」顯示畫面。

```
                    啓動系統服務工具 (STRSST) 登入

                                                        系統：RCHSYS01
請鍵入選項，按 Enter 鍵。

   服務工具使用者 . . . . . _______
   服務工具密碼 . . . . . . _______









F3=跳出        F9=變更密碼        F12=取消
```

輸入服務工具使用者設定檔名稱和密碼。應顯示「系統服務工具」顯示畫面。

```
                        系統服務工具 (SST)

 請選取下列其中一項：

      1. 啓動服務工具
      2. 使用作用中的服務工具
      3. 使用硬碟機
      4. 使用磁片資料回復
      5. 使用系統分割區
      6. 使用系統容量




 選項
     1
 F3=跳出        F10=輸入指令          F12=取消
```

請選取 **1** 以啓動服務工具並按下 Enter 鍵。將顯示「啓動服務工具」顯示畫面。

```
                          啓動服務工具

警告：不正確地使用此服務工具可導致此系統中的資料損壞。
請聯絡您的客戶服務代表以獲得援助。


請選取下列其中一項：

     1. 產品活動日誌
     2. 追蹤授權內碼
     3. 使用通信追蹤
     4. 顯示/變更/傾出
     5. 授權內碼日誌
     6. 主記憶體傾出管理程式
     7. 硬體服務管理程式



選項
     7

F3=跳出           F12=取消          F16=SST 功能表
```

選取 **7** 以啓動「硬體服務管理程式」。將顯示「硬體服務管理程式」螢幕，顯示可用選項的功能表。

```
                       硬體服務管理程式

注意：此公用程式僅提供給客戶服務代表使用。

   系統裝置 . . . . . :  9406-270 10-4314M
   版次 . . . . . :  V5R1M0 (1)

請選取下列其中一項：

   1. 包裝硬體資源 (系統、框架、卡...)
   2. 邏輯硬體資源 (匯流排、IOP、控制器...)
   3. 按資源名稱尋找資源
   4. 失敗及未報告的硬體資源
   5. 系統電源控制網路 (SPCN)
   6. 使用服務動作日誌
   7. 顯示標籤位置工作底稿
   8. 裝置並行維護

選項
     2

F3=跳出       F6=列印配置        F9=顯示卡之間隙資訊
F10=顯示需要注意的資源       F12=取消
```

選取 **2** 以使用邏輯硬體資源。

```
                              邏輯硬體資源

請選取下列其中一項：

     1. 系統匯流排資源
     2. 處理器資源
     3. 主記憶體資源
     4. 高速線路鏈結資源









選項
     1

F3=跳出          F6=列印配置          F12=取消
```

由「邏輯硬體資源」螢幕上選取 **1** 以顯示系統匯流排資源。

```
                        系統匯流排上的邏輯硬體資源

要使用的系統匯流排 . . . . . . . . . *ALL *ALL, *SPD, *PCI, 1-511
按子集 . . . . . . . . . . . . . . . *CRP *ALL, *STG, *WS, *CMN, *CRP

請鍵入選項，然後按 Enter 鍵。
  2=變更明細      4=移除     5=顯示明細     6=I/O 除錯
  8=相關的包裝資源       9=與 IOP 相關的資源

                                                                資源
Opt 說明                                  機型        狀態       名稱
 _ HSL I/O 橋接器                         2249-       作業       BC02
 _   匯流排擴充配接卡                     -           作業       BCC02
 _    系統匯流排                          2249-       作業       LB01
 _     多重配接卡橋接器                   2249-       作業       PCI01D
 _      組合的功能 IOP * <          284D-001    作業          CMB01
 _ HSL I/O 橋接器                         283B-       作業       BC01
 _   匯流排擴充配接卡                     -           作業       BCC03
                                                                尚有...




F3=跳出     F5=重整    F6=列印     F8=併入未報告資源
F9=失敗的資源       F10=未報告資源
F11=顯示序列/組件號          F12=取消
```

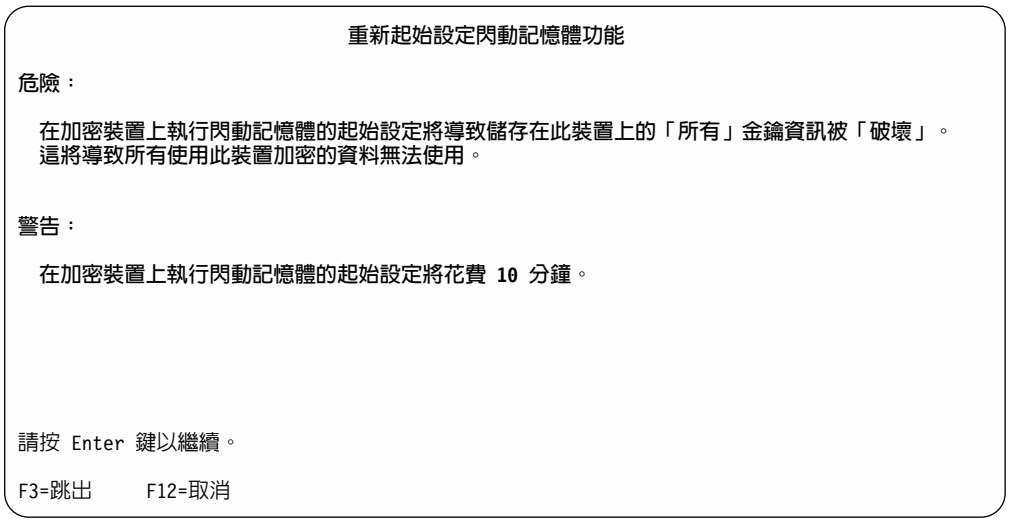
如果您知道哪個 IOP 包含 4758，請在該 IOP 旁邊鍵入 **9**。否則，請在「按子集」欄位鍵入 *CRP 來子集化此清單，然後在包含 4758 的 IOP 旁邊鍵入 **9**。然後請參閱與 IOP 顯示相關的「邏輯硬體資源」。

```
                與 IOP 相關的邏輯硬體資源

請鍵入選項，然後按 Enter 鍵。
   2=變更明細      4=移除     5=顯示明細     6=I/O 除錯
   7=驗證          8=相關的包裝資源

                                                                資源
Opt 說明                                機型        狀態        名稱
  _   組合功能 IOP  * <  284D-001  作業         CMB01
  _   加密配接卡            4758-023  作業         CRPCTL01
  6   加密裝置   758-023   作業        CRP01
  _   工作站 IOA            2746-001   作業        CTL01
  _   顯示站                3477-0FC   作業        DSP001
  _   顯示站                3477-0FC   作業        DSP002
  _   通信 IOA              2745-001   作業        LIN01
  _   通信埠                2745-001   作業        CMN01
  _   通信埠                2745-001   作業        CMN02
  _   通信 IOA              2744-001   作業        LIN03
  _   通信埠                2744-001   作業        CMN03
                                                                尚有...
F3=跳出     F5=重整    F6=列印     F8=併入未報告資源
F9=失敗的資源      F10=未報告資源
F11=顯示序列/組件號         F12=取消
```

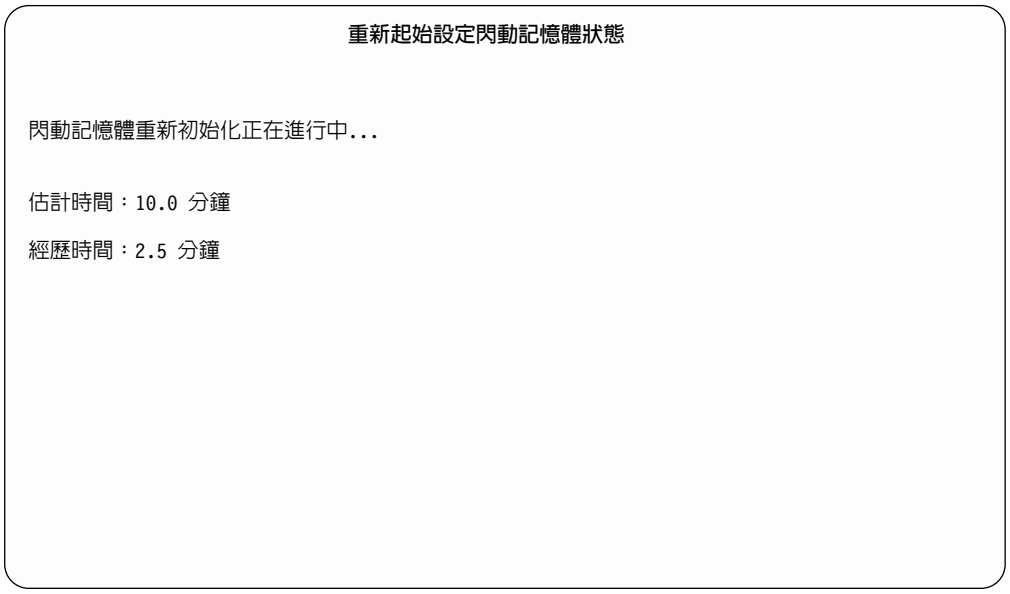
在需要重新起始設定的加密裝置後面鍵入 **6**，然後按 Enter 鍵。

```
                    選取加密除錯功能

請選取下列其中一項：

      1. 重新起始設定閃動記憶體
      2. 選取 IOP 除錯功能

選項
     1

F3=跳出      F12=取消
```

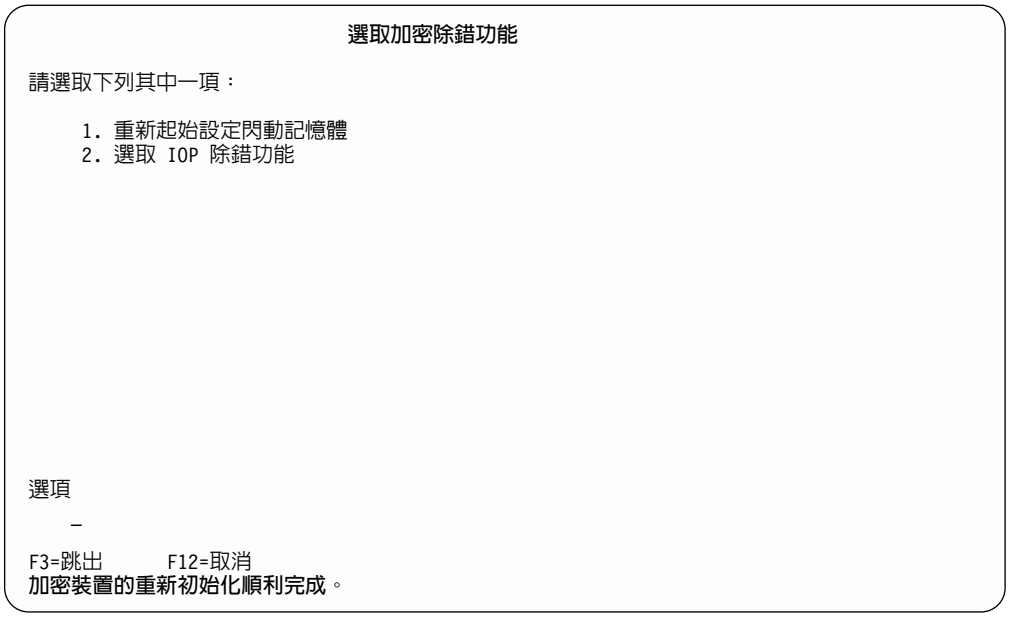
選取 **1**，以重新起始設定閃動記憶體 (重新載入「4758 輔助處理器授權內碼」)。 將會顯示一個確認螢幕。若您正在套用 PTF，請確定對加密的資料和金鑰已經採取了必需的預防措施，並且備份了主要金鑰。請按 Enter 鍵以繼續。



下列顯示重新初始化狀態的螢幕將被顯示並更新，直到完成重新初始化。



當完成重新初始化時，將顯示一個訊息。



在完成重新初始化之後，在每一個必需的螢幕中按功能鍵 F3，可完全退出系統服務工具。

第 5 章 加密硬體的相關資訊

下列資源提供了關於加密概念或硬體的其餘資訊：

IBM 紅皮書™

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

IBM 來源

- IBM 加密硬體 包含 iSeries 伺服器的 「4758 加密輔助處理器」硬體解決方案的相關資訊之網站。
- CCA Basic Services Manual 是爲評估或建立「IBM 4758 共同密碼架構 (CCA)」支援程式的系統及應用程式分析師與應用程式設計師準備的。
- IBM PCI Cryptographic Coprocessor 文件檔案庫 包含了包括「4758 加密輔助處理器」的一般、支援及程式設計資訊之可下載 PDF 文件。
- IBM developerWorks 包括了加密概念、加密法及加密的指導教學的網站。

© Copyright IBM Corp. 1998, 2003

第 6 章 程式碼不保事項聲明

此文件包含程式設計範例。

IBM 授與您使用所有程式設計程式碼範例的非專屬授權，您可以依據這些範例，產生類似的函數，來符合您的需要。

IBM 提供的所有範例程式碼僅做爲說明用途。這些範例尚未徹底經過所有情況的測試。因此 IBM 不擔保或默示保証這些程式的可靠性、可用性或功能。

所有內含於此的程式是以「現況」提供給您，不具任何形式的擔保。IBM 明示排除有關這些程式的不侵權、可售性、符合特定使用目的之默示擔保。

© Copyright IBM Corp. 1998, 2003