

IBM 4758 PCI Cryptographic Coprocessor

CCA Support Program Installation Manual
for IBM 4758 Models 002 and 023
Release 2.41

Manual Revised August, 2002



Note!

Before using this information and the product it supports, be sure to read the general information printed under "Notices" on page X-1.

b Twelfth Edition (August, 2002)

This edition applies to:

- IBM 4758 Models 002 and 023 (and not to Models 001 and 013).
- 1 • Release 2.4.1.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for IBM AIX.*
- 1 • Release 2.41 of the licensed CCA Cryptographic Coprocessor Support Program feature for Microsoft Windows NT** and
- 1 Windows 2000.**

Each feature is designed to provide software support for an IBM 4758 PCI Cryptographic Coprocessor, Model 002 or Model 023, installed into a computer running the accompanying operating system.

Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, please check the IBM 4758 product Web site for an updated version of this publication.

IBM does not stock publications at the address given below. This and other publications related to the IBM 4758 Coprocessor can be obtained in PDF format from the Library page at <http://www.ibm.com/security/cryptocards>.

Readers' comments can be communicated to IBM by following the instructions provided in the "Contact the Crypto team!" section on the product Web site at <http://www.ibm.com/security/cryptocards>, or you can respond by mail to the following address:

Department VM9A, MG81/204-3
IBM Corporation
8501 IBM Drive
Charlotte, NC 28262-8563
USA

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Several different revision codes may appear in this book and indicate items that have changed in:

- 1 • The eleventh edition with the initial revisions for Release 2.41
- b • The twelfth edition in August, 2002, also covering Release 2.41.

© Copyright International Business Machines Corporation 1997-2002. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

About This Publication	ix
Audience	ix
Prerequisite Knowledge	ix
Organization of This Publication	x
Related Publications	x
IBM 4758 PCI Cryptographic Coprocessor Publications	x
Summary of Changes	xi
Chapter 1. Installation Process Overview	1-1
Summary	1-1
Chapter 2. Obtaining Coprocessor Hardware and Software	2-1
How to Choose, Order, and Install Coprocessor Hardware	2-1
Ordering Coprocessors for Windows Systems	2-1
Ordering Coprocessors for pSeries Servers	2-2
Replacement Batteries	2-2
How to Place Orders for the IBM 4758 Coprocessor	2-2
How to Install Your IBM 4758 Hardware	2-2
How to Download the Software	2-3
Chapter 3. Installing the Support Program	3-1
Support Program Components	3-1
How to Install and Remove Coprocessor Host Software	3-1
How to Install and Remove the Support Program for AIX	3-2
AIX Requirements	3-2
How to Install the Support Program Base Release 2.4.1.0	3-2
How to Configure the Support Program	3-3
CCA Support Program and AIX File Permissions	3-4
Updating from a Prior Release	3-5
How to Review pSeries (RS/6000) Coprocessor Hardware Errors	3-5
How to Remove the Support Program	3-6
How to Install and Remove the Support Program for Windows	3-7
Windows NT and Windows 2000 Requirements	3-7
How to Install the Support Program	3-8
How to Configure the Support Program	3-10
How to Remove the Support Program	3-10
What to Do When Migrating from Windows NT to Windows 2000	3-11
Chapter 4. Loading Software into the Coprocessor	4-1
How to Load Coprocessor Software	4-1
Changing the Default Directory and Running CLU	4-2
Determining Coprocessor Software Segment Contents	4-3
Changing Software Segment Contents	4-4
Validating the Coprocessor Segment Contents	4-5
How to Unload Coprocessor Software and Zeroize the CCA Node	4-5
Coprocessor Load Utility Reference	4-6
Coprocessor Memory Segments	4-6
Validation of Coprocessor Software Loads	4-7
Coprocessor Load Utility Syntax	4-8
Coprocessor Load Utility Commands	4-9

Coprocessor Load Utility Return Codes	4-11
Chapter 5. Using the CNM and CNI Utilities to Manage the Cryptographic Node	5-1
Overview	5-1
Cryptographic Node Management Utility Overview	5-2
Cryptographic Node Initialization Utility Overview	5-2
How to Use the Utilities, Sample Scenarios	5-3
How to Establish a Test Node	5-3
How to Establish Nodes in a Production Environment	5-4
Access-Control-Administrator Procedure	5-5
Key-Management-Officer Procedures	5-6
How to Use the CNM Administrative Functions	5-7
How to Choose a Coprocessor	5-7
How to Initialize (Zeroize) the Node	5-7
How to Log On and Off the Node	5-8
How to Load the Function-Control Vector	5-8
How to Configure the Cryptographic Node Management Utility	5-8
How to Synchronize the Clock-Calendar	5-8
How to Obtain Status Information	5-9
How to Create and Manage Access-Control Data	5-9
Access-Control Overview	5-10
Initial State of the Access-Control System	5-10
How to Define a Role	5-11
How to Edit Existing Roles	5-12
How to Edit a Disk-Stored Role	5-12
How to Edit a Coprocessor-Stored Role	5-12
How to Delete a Coprocessor-Stored Role	5-12
How to Define a User Profile	5-13
How to Edit Existing User Profiles	5-14
How to Edit a Disk-Stored User Profile	5-14
How to Edit a Coprocessor-Stored User Profile	5-14
How to Delete a Coprocessor-Stored User Profile	5-14
How to Reset the User-Profile-Failure Count	5-15
How to Initialize the Access-Control System	5-15
How to Manage Cryptographic Keys	5-15
How to Manage the Master Key	5-16
How to Verify an Existing Master Key	5-16
How to Auto-Set or Randomly Generate the Master Key	5-17
How to Load a New Master-Key from Key Parts	5-17
How to Clone a Master Key	5-18
Managing Key Storage	5-21
How to Create or Initialize Key Storage	5-21
How to Reencipher Stored Keys	5-22
How to Delete a Stored Key	5-22
How to Create a Key Label	5-23
How to Create and Store Primary DES KEKs	5-23
Using the CNI Utility to Establish Other Nodes	5-24
Chapter 6. Observations on Secure Operations	6-1
Ensuring Code Levels Match and IBM CCA Code is Installed	6-1
Access Controls	6-1
Locking the Access-Control System	6-2
Passphrase Considerations	6-2

	Roles and Profiles	6-3
	Cryptographic Keys	6-4
	CCA Asymmetric DES keys	6-4
	Clear Key-Parts	6-6
	Key Export	6-7
	Unwrapping Keys	6-7
	Operations with Clear Keys	6-7
b	Using DES Replicated Keys	6-8
b	Generating RSA Keys	6-9
b	Regeneration Data	6-9
b	Attributes for Signature and Key Management	6-9
b	Retained Keys	6-9
	PIN Data	6-10
	Status Data	6-10
	RS-232 Port	6-10
	Master-Key Cloning	6-10
	Sample Access-Control Regimes	6-11
b	Digital-Signing Server	6-12
b	Application Environment	6-12
b	Threat Considerations	6-13
b	Security Objectives	6-13
b	Application Requirements	6-14
b	Policy Requirements	6-14
b	Operational Requirements and Plan	6-16
1	Secured Code-Signing Node	6-21
	 Chapter 7. Building Applications to Use with the CCA API	7-1
	Overview	7-1
	How to Call Verbs in C Program Syntax	7-1
	How to Compile and Link Application Programs	7-2
	Compiling Applications for AIX	7-2
	Sample Routine	7-2
	Enhancing Throughput with CCA and the 4758 Models 002 and 023	7-8
	Multi-threading and Multi-processing	7-8
	Caching DES and RSA Keys	7-9
	 Appendix A. CCA Access-Control Commands	A-1
	 Appendix B. Initial DEFAULT-Role Commands	B-1
	 Appendix C. Machine-Readable-Log Contents	C-1
	 Appendix D. Migration Considerations, Version 1 to 2	D-1
	 Appendix E. Device Driver Error Codes	E-1
	 Appendix F. Master-Key Cloning	F-1
	Master-Key Cloning Procedure	F-1
	Phase 1: Establish the Share Administration Node	F-5
	Phase 2: Establish the Source Node	F-6
	Phase 3: Establish Target Node and Clone Master Key	F-7
	Access-Control Considerations when Cloning	F-9
b	 Appendix G. Threat Considerations for a Digital-Signing Server	G-1

Notices	X-1
License	X-1
Copying and Distributing Softcopy Files	X-1
Trademarks	X-2
List of Abbreviations and Acronyms	X-3
Glossary	X-5
Index	X-11

Figures

1

4-1.	Typical CLU Status Response	4-3
4-2.	Typical CLU System Status Response	4-10
5-1.	The Role Definition Panel	5-11
5-2.	The Profile Management Panel	5-13
5-3.	The Load Master Key Panel	5-17
5-4.	The CCA Node Initialization Editor Panel	5-24
7-1.	Syntax, Sample Routine	7-4
F-1.	Cloning Responsibilities, Profiles and Roles	F-3
F-2.	Cloning-Information Worksheet	F-4

About This Publication

1

This installation and operation guide describes Release 2.41 (AIX: 2.4.1.0) of the IBM 4758 CCA Support Program for the IBM 4758 PCI Cryptographic Coprocessor, Models 002 and 023. The Support Program includes device drivers, utilities, and the IBM Common Cryptographic Architecture (CCA) Coprocessor code.

You can obtain Support Program editions to use with AIX, Windows NT, and Windows 2000 operating systems.

Note: The OS/2 platform is not supported by this release.

Use this manual to help with the following tasks:

- Obtain the Support Program through the Internet
- Load the software onto a host computer and into the Coprocessor(s)
- Use the utilities supplied with the Support Program to:
 - Load the Coprocessor function-control vector (FCV)
 - Initialize one or more Coprocessors
 - Create and manage access-control data
 - Create a master key and primary key-encrypting keys (KEKs)
 - Manage key storage at the cryptographic node
 - Create node-initialization file lists to set up and configure other cryptographic nodes
- Link your application software to the CCA libraries
- Obtain guidance for security consideration in application development and operational practices.

Audience

The audience for this publication includes:

- System administrators who install the software
- Security officers responsible for the Coprocessor access-control system
- System programmers and application programmers who determine how the software is to be used.

Prerequisite Knowledge

Before you use this publication, familiarize yourself with the contents of the *IBM 4758 PCI Cryptographic Coprocessor General Information Manual*, available as a PDF file on the Library page of the IBM 4758 Web site, <http://www.ibm.com/security/cryptocards>. This manual describes the IBM 4758 PCI Cryptographic Coprocessor hardware and the CCA Cryptographic Coprocessor Support Program feature.

Organization of This Publication

- Chapter 1, "Installation Process Overview" summarizes the installation and the operation of the CCA Cryptographic Coprocessor Support Program
- Chapter 2, "Obtaining Coprocessor Hardware and Software" describes how to obtain the PCI cryptographic Coprocessor hardware and the CCA Cryptographic Coprocessor Support Program
- Chapter 3, "Installing the Support Program" describes how to install the software onto the host computer
- Chapter 4, "Loading Software into the Coprocessor" describes how to load the operating system and the CCA software into the PCI Cryptographic Coprocessor
- Chapter 5, "Using the CNM and CNI Utilities to Manage the Cryptographic Node" describes how to use the Cryptographic Node Management and the Cryptographic Node Initialization utilities to set up and manage cryptographic nodes
- Chapter 6, "Observations on Secure Operations" offers guidance in operating the CCA implementation with increased security
- Chapter 7, "Building Applications to Use with the CCA API" explains how to build applications for CCA, and how to link them to CCA libraries
- Appendix A, "CCA Access-Control Commands" lists the commands used by the CCA API as it requests service from the PCI Cryptographic Coprocessor
- Appendix B, "Initial DEFAULT-Role Commands" details the permissions granted to the DEFAULT role when the access-control system is initialized
- Appendix C, "Machine-Readable-Log Contents" details the content of the machine readable log created by the Coprocessor Load Utility
- Appendix D, "Migration Considerations, Version 1 to 2" provides guidance concerning migration for CCA customers from IBM 4758-001/013 to IBM 4758-002/023
- Appendix E, "Device Driver Error Codes" provides error-code information that can be observed when operating the CLU utility
- Appendix F, "Master-Key Cloning" provides a procedure for master-key cloning
- Appendix G, "Threat Considerations for a Digital-Signing Server" and other application areas
- Product and publication notices, a list of abbreviations, a glossary, and an index complete the manual.

b
b

Related Publications

The list below reflects source information regarding the PCI Cryptographic Coprocessor and commercial cryptographic applications in general. Publications regarding other IBM cryptographic products that utilize the CCA application program interface (API) are listed in the *IBM 4758 PCI Cryptographic Coprocessor General Information Manual*.

IBM 4758 PCI Cryptographic Coprocessor Publications

For availability of these publications, check the Library page of the product Web site at <http://www.ibm.com/security/cryptocards>. From the Web site, you can download, view, and print publications available in the Adobe Acrobat** portable document format (PDF):

- *IBM 4758 CCA Basic Services Reference and Guide*
- *IBM 4758 PCI Cryptographic Coprocessor General Information Manual*
- *IBM 4758 PCI Cryptographic Coprocessor Installation Manual*
- AIX users should also obtain and reference the *PCI Cryptographic Coprocessor Installation and Using Guide*.

Summary of Changes

Changes in this manual...

b
b
b
b
b
b
b
b
b
b
b
b
b
b
b
b

Release 2.41 (2.4.1.0), Revised Manual August, 2002

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains these documentation updates:

- Added advice concerning retaining key-storage in an AIX Support Program update. See “Updating from a Prior Release” on page 3-5.
- Added mention that Windows 2000 Advanced Server is supported.
- Corrected the “SET TZ” example in “How to Install the Support Program” on page 3-8.
- Updates to the list of Coprocessor “class files.” See “Validating the Coprocessor Segment Contents” on page 4-5.
- Added advice in “Generating RSA Keys” on page 6-9.
- Added advice concerning how to set up a “Digital-Signing Server” on page 6-12.
- Appendix G, “Threat Considerations for a Digital-Signing Server” discusses items which have been addressed in the design and implementation of the Support Program and Coprocessor, and items which you should consider in your deployment of the Coprocessor. While the appendix is specific to the case of a Digital-Signing Server, you will find this material of interest in any application of the Coprocessor.
- Updated return code 8340xxxx and added 8340FFDF in Appendix E, “Device Driver Error Codes.”

1
1
1
1
1
1
1
1
1
1
1
1

Release 2.41 (2.4.1.0)

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains product information that is current with:

- Release 2.4.1.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for AIX
- Release 2.41 of the licensed CCA Cryptographic Coprocessor Support Program feature for Windows NT and Windows 2000.

Except for Chapter 6, the manual contains only minor changes for Release 2.41. Note that with Release 2.41 AIX users have a choice of drivers to download for AIX Release 4.3.3 or Release 5.1.

Release 2.40 (2.4.0.0)

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains product information that is current with:

- Release 2.4.0.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for AIX
- Release 2.40 of the licensed CCA Cryptographic Coprocessor Support Program feature for Windows NT and Windows 2000.

The manual contains no major new or changed information for Release 2.40.

Release 2.31 (2.3.1.0)

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains product information that is current with:

- Release 2.3.1.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for AIX
- Release 2.31 of the licensed CCA Cryptographic Coprocessor Support Program feature for OS/2
- Release 2.31 of the licensed CCA Cryptographic Coprocessor Support Program feature for Windows NT and Windows 2000.

Appendix F, "Master-Key Cloning" is added to the manual. No other major changes are incorporated.

Release 2.30 (2.3.0.0)

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains product information that is current with:

- Release 2.3.0.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for AIX
- Release 2.30 of the licensed CCA Cryptographic Coprocessor Support Program feature for OS/2
- Release 2.30 of the licensed CCA Cryptographic Coprocessor Support Program feature for Windows NT and Windows 2000.

The manual contains these major changes for Release 2.30:

- Support for AIX (this was first available for 2.2.1.0 in October, 2000).
- Support for Windows 2000.
- CCA support for multiple Coprocessors.
- Modifications on the Coprocessor Load Utility (CLU) status report and addition of the "SS" command to report on all installed Coprocessors.
- Inclusion of the Function Control Vector (FCV) in each individual distribution without the need to specify a specific feature code for the level of FCV. All customers are now entitled to use an RSA key length of 1024 bits when performing key management.
- Appendix F, "Master-Key Cloning" is added for additional guidance on cloning a master key.

Note: Version 2 (Release 2.30, 2.3.0.0) does not support IBM 4758 models 001 and 013.

Release 2.20/2.21

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains product information that is current with:

- Release 2.2.1.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for AIX
- Release 2.20 of the licensed CCA Cryptographic Coprocessor Support Program feature for OS/2
- Release 2.20 of the licensed CCA Cryptographic Coprocessor Support Program feature for Windows NT.

This edition also incorporates new procedures for downloading of the CCA Support Programs. You no longer need to obtain License Keys to decrypt the downloaded software. When downloading software, you will need to complete a registration process.

Release 2.20

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains product information that is current with:

- Release 2.20 of the licensed CCA Cryptographic Coprocessor Support Program feature for OS/2
- Release 2.20 of the licensed CCA Cryptographic Coprocessor Support Program feature for Windows NT.

Release 2.20 provides:

- Support of the IBM 4758 Models 002 and 023
- Additional function including triple-DES encryption for general data.

Notes:

1. Release 2.20 does **not** support IBM 4758 Models 001 and 013.
2. Logon to a profile has been supported from a thread in Release 1.31; this support is continued in Release 2.20. See "How to Log On and Off the Node" on page 5-8.

Chapter 1. Installation Process Overview

This chapter summarizes the installation and operation procedures discussed in this manual and provides a checklist for you to use while installing the PCI Cryptographic Coprocessor and the CCA Cryptographic Coprocessor Support Program. See Table 1-1 on page 1-2.

Summary

The CCA Cryptographic Coprocessor Support Program consists of several components, and includes:

- Device drivers and an operating system for the PCI Cryptographic Coprocessor hardware
- Support for the IBM Common Cryptographic Architecture (CCA) application program interface (API)
- A function-control vector
- Utility applications that run on the host RS/6000 machine or on the personal computer (PC) into which the Coprocessor has been installed.

A function-control vector is a signed value provided by IBM. Its use originated to enable the CCA application within the Coprocessor to yield a level of cryptographic service consistent with applicable cryptographic implementation import and export regulations.

b
b

See the AIX or Windows portions of Chapter 3, "Installing the Support Program" for the supported operating system versions and prerequisite software.

To install these components and to establish a CCA cryptographic node, perform the following steps described in this manual:

1. **Obtain the Hardware and Software:** Chapter 2, "Obtaining Coprocessor Hardware and Software" describes how to order the hardware from IBM, how to download the software through the Internet, and how to unpack the downloaded files.

Note: Customers no longer need to obtain License Keys, but instead are guided through a registration process prior to downloading the code. This process is described in "How to Download the Software" on page 2-3. Also, the downloaded code is no longer encrypted.

2. **Install the Software onto the Host:** Chapter 3, "Installing the Support Program" describes how to install the downloaded software onto the Coprocessor host computer.
3. **Load the Coprocessor Software:** Chapter 4, "Loading Software into the Coprocessor" describes how to load both the CP/Q++ embedded operating system, and the CCA application program.

4. **Set Up the Cryptographic Node:** You can establish a CCA cryptographic node using the utilities provided with the support program, or by linking your application programs to the CCA API. You should also verify the access control and other setup requirements imposed by application software you plan to use with the IBM 4758. The Cryptographic Node Management utility described in Chapter 5, "Using the CNM and CNI Utilities to Manage the Cryptographic Node" includes setup and management functions needed to:

- Load the Function-Control Vector
- Create and edit the access-control data
- Manage the Coprocessor master key
- Manage primary KEKs
- Manage the storage of data keys
- Create lists ("scripts") for the Cryptographic Node Initialization utility.

5. **Link Application Programs to the CCA Libraries:** Chapter 7, "Building Applications to Use with the CCA API" describes how to build applications for CCA and how to link them to the CCA libraries.

Table 1-1. Activity Checklist, CCA Cryptographic Coprocessor Support Program Installation

Step	Task	Reference	√
1	Decide which platform support packages are appropriate to your setup: AIX () Windows NT () Windows 2000 () Windows Advanced Server ()	"How to Choose, Order, and Install Coprocessor Hardware" on page 2-1	
2	Place an order with IBM or your IBM Business Partner. (OEM sales are processed by the IBM OEM Sales office.)	"How to Place Orders for the IBM 4758 Coprocessor" on page 2-2	
3	Receive Coprocessor hardware.		
4	Install Coprocessor hardware. "How to Install Your IBM 4758 Hardware" on page 2-2 discusses which must be installed first, the hardware or the device driver.	"How to Install Your IBM 4758 Hardware" on page 2-2	
5	Download the support program for your operating system.	"How to Download the Software" on page 2-3	
6	Install the support program onto the Coprocessor host computer.	Chapter 3, "Installing the Support Program"	
7	Load Coprocessor software.	Chapter 4, "Loading Software into the Coprocessor"	
8	Set up a CCA test node. Review the first pages in Chapter 5, "Using the CNM and CNI Utilities to Manage the Cryptographic Node." Then set up a test node.	"How to Establish a Test Node" on page 5-3	
9	Run test programs that utilize the CCA libraries.		

b
b
b
b

Chapter 2. Obtaining Coprocessor Hardware and Software

The CCA Cryptographic Coprocessor Support Program feature is available for download through the Internet on the Order page at <http://www.ibm.com/security/cryptocards>. This chapter describes:

- How to choose, order, and install Coprocessor hardware
- How to Download the Software.

How to Choose, Order, and Install Coprocessor Hardware

The Coprocessor is manufactured in several models, each with different capabilities. Models 002 and 023 incorporate triple-DES and faster hardware than the earlier models 001 and 013. Only models 002 and 023 will operate with the Release 2.xx CCA software. Model 002 includes advanced physical penetration detection and the product has been certified under FIPS 140-1 at level 4. The Model 023 incorporates a different approach to physical penetration detection but in other respects the same as the Model 002. The Model 023 is certified under FIPS 140-1 at level 3.

IBM manufactures two variations of both the Models 002 and 023. The first variations incorporate two batteries, operate on a 5.0 volt PCI bus, and are supplied when you order an IBM 4758 “machine type.” The second variations incorporate four batteries, operate on 3.3 or 5.0 volt PCI bus systems, and are supplied when you order features for IBM eServer iSeries, pSeries, and zSeries server systems.

Subsequent sections cover:

- Ordering Coprocessors for Windows systems
- Ordering Coprocessors for pSeries (RS/6000) servers
- Replacement batteries
- How to place orders for the IBM 4758 hardware
- How to install your IBM 4758 hardware.

1

Ordering Coprocessors for Windows Systems

IBM 4758 Model 002 (FIPS 140 level 4) and IBM 4758 Model 023 (FIPS 140 level 3) are ordered from IBM as a machine-type and model. The Coprocessor can be installed in typical “Wintel” server and desk-top systems such as the IBM eServer xSeries machines. The Coprocessor requires PCI slots that accept full-height, two-thirds-length, PCI boards. IBM tests the Coprocessor in most of the IBM xSeries servers. Due to the large number and variety of systems in the marketplace, IBM does not test the Coprocessor in other machines. If you encounter difficulty, contact IBM via the <http://www.ibm.com/security/cryptocards> Web site. IBM will endeavor to assist in problem determination and resolution, but makes no commitment that problems with other system types can be resolved.

The software support for the Windows NT and Windows 2000 will support up to eight Coprocessors per system.

Ordering Coprocessors for pSeries Servers

IBM eServer pSeries (RS/6000) users order the Coprocessor as a “feature” when they order their system. The following feature code is offered:

4963 Model 002 class technology FIPS 140 level 4.

pSeries literature explains which systems support installation of the Coprocessor and any limitations. A summary of this information can be found on the IBM eServer pSeries page on the <http://www.ibm.com/security/cryptocards> Web site.

Note: pSeries Engineering does not support use of the IBM 4758 (two-battery, 5.0 volt) Model 002 or Model 023 Coprocessors in pSeries systems.

Replacement Batteries

IBM offers a replacement-battery kit so that you can maintain the functionality of the Coprocessor. The batteries provide power to a small quantity of internal memory, the clock-calendar, the tamper-detection circuitry, and so forth. It is imperative that batteries with sufficient stored-energy power the Coprocessor when it is not in a powered-on system. In the event that the batteries fail, or are removed from the Coprocessor, the unit will zeroize and be rendered permanently inoperable. There is no recovery from this situation.

The battery kit contains two batteries and a temporary-battery tray. The shelf life of the batteries in the kit is nearly the same as the useful life of batteries mounted in an IBM 4758 that is continuously powered on. A battery kit should be ordered and the batteries changed as a planned maintenance activity every 3 to 5 years. The actual life of the batteries is anticipated to be in excess of 5 years. When you do change batteries, be sure that they are fresh and have not been in inventory for a long period. pSeries users should order two battery kits for a total of four batteries per Coprocessor.

The battery kit is ordered as feature code 1008.

How to Place Orders for the IBM 4758 Coprocessor

To order the Coprocessor hardware, contact your local IBM Representative or your IBM Business Partner, and order the models and features you have chosen.

Customers in the U.S.A. can also contact IBM Direct at 1-800-IBM-CALL. Specifically mention “IBM 4758” so that you can discuss your order with the group that processes IBM 4758 orders.

How to Install Your IBM 4758 Hardware

The IBM 4758 is installed in a manner similar to other PCI boards.

- Personal computer users should follow the process described in the *IBM 4758 PCI Cryptographic Coprocessor Installation Manual*.
- pSeries users should follow the process described in the *PCI Cryptographic Coprocessor Installation and Using Guide*. (This book can be obtained in Adobe PDF format from http://www.rs6000.ibm.com/resource/hardware_docs. Find “PCI Cryptographic” in the list of books.) Note that the order of installation between the hardware and the device driver is important in an AIX installation; follow the guidance in the book.

Be certain that you never remove the Coprocessor batteries except as outlined in the battery-replacement procedure in the *IBM 4758 PCI Cryptographic Coprocessor Installation Manual* or, for pSeries users, the *PCI Cryptographic Coprocessor Installation and Using Guide*. The Coprocessor is certified at the factory. If it ever detects tampering, or if both battery power and system power are simultaneously removed, the factory certification is zeroized and the Coprocessor is rendered non-functional. There is no recovery from this situation.

It is possible to inadvertently cause a tamper event if you cause some of the Coprocessor circuitry to short-circuit. Remember that the batteries on the Coprocessor supply power to tamper sensors. If in handling the Coprocessor you cause a short circuit in this circuitry, this could result in a tamper event. This is very unlikely to occur, but you should be careful when installing the Coprocessor to keep the circuitry on the board from contacting conductive portions of the host machine or adjacent boards.

How to Download the Software

You download the support program software through the Internet.

Tip: To be sure you receive the latest version of the support program, wait to download the software until you have received your Coprocessor. At that time you should also check the Web site for any available fixes. See the Software Updates section of the IBM 4758 product Web site at <http://www.ibm.com/security/cryptocards>.

Select the software that you require by operating-system platform, release level, and support program as indicated on the Software Download page of the product Web site, <http://www.ibm.com/security/cryptocards>. You are prompted to complete a registration procedure, then you are presented with a page from which you choose items to download.

pSeries users should download both the CCA-PKCS#11 Common Support and the CCA Support. The Cryptographic Coprocessor device driver for AIX can also be downloaded if it is at a higher level than already installed on your system or available on the AIX system CD. These three items together comprise the CCA Support Program for AIX.

If you plan to use the support program on multiple host computers, you can copy the downloaded files to the other hosts.

Now you are able to install the support program; see Chapter 3, “Installing the Support Program.”

Chapter 3. Installing the Support Program

After downloading the software as described in Chapter 2, "Obtaining Coprocessor Hardware and Software," follow the procedures in this chapter to install the CCA Cryptographic Coprocessor Support Program onto the Coprocessor host computer. (Loading software into the Coprocessor is described in Chapter 4, "Loading Software into the Coprocessor" and initializing the CCA application within the Coprocessor is described in Chapter 5, "Using the CNM and CNI Utilities to Manage the Cryptographic Node.")

This chapter:

- Lists the support program components you are installing
- Lists system prerequisites to installing the software
- Describes how to install the software
- Describes how to remove the software
- Describes what to do when migrating from Windows NT to Windows 2000.

Support Program Components

The procedures in this chapter install the following support program components onto the host computer:

- Device drivers for the IBM 4758 PCI Cryptographic Coprocessor
- The shared libraries or DLLs necessary to link the CCA application program interface (API) to the Coprocessor driver
- The Coprocessor load utility and software files necessary to load the CP/Q++ operating system and the CCA application program into the Coprocessor; the utility is described in Chapter 4, "Loading Software into the Coprocessor"
- The Cryptographic Node Management utility necessary to load the function-control vector (FCV) into the Coprocessor and to set up a cryptographic node; the utility is described in Chapter 5, "Using the CNM and CNI Utilities to Manage the Cryptographic Node."

How to Install and Remove Coprocessor Host Software

For each operating system, the following sections:

- List the requirements for the support program
- Describe how to install the support program
- Describe how to remove the support program.

After you have installed the platform-specific software as described in this chapter, you are ready to install software into the Coprocessor; see Chapter 4, "Loading Software into the Coprocessor."

Note that all components of the installed CCA package may not indicate the same version number. The version number for a software component is only updated when the component is changed for a release. Unchanged components retain the version number from the last time they were modified. For example, the Cryptographic Coprocessor Support Software may be at a different level from the Coprocessor device driver.

How to Install and Remove the Support Program for AIX

This section includes a description of the support program system requirements and procedures necessary to install and remove the *base release 2.4.1.0* software. See “Updating from a Prior Release” on page 3-5 if you are updating from a prior release.

Important: The installation process requires root-level authority; it must be performed by a system administrator with that authority.

AIX Requirements

Before you install the support program, make sure your system meets the following requirements:

Hardware:

An IBM eServer pSeries (RS/6000) server with an available PCI Cryptographic Coprocessor feature.

During installation of the software, the driver interacts with the Coprocessor to arbitrate interrupt settings, DMA channels, and other system resources. For installation instructions regarding the Coprocessor hardware and device driver, refer to the *PCI Cryptographic Coprocessor Installation and Using Guide*.

Software:

AIX Version 4.3.3 (32-bit mode only) or 5.1

Java** runtime environment 1.1.8 through 1.3 (it is possible that a later release of Java will work, but it has not been tested by IBM), available from <http://www.ibm.com/java/jdk/download>. This is required to use the Cryptographic Node Management utility.

These software packages which you download from the IBM 4758 Web site, <http://www.ibm.com/security/cryptocards>:

- Release 2.4.1.0 CCA-PKCS#11 Common Support, file **csuf.cca.bff**.
- Release 2.4.1.0 CCA Support, file **csuf.com.bff**.
- A device driver appropriate to your release of AIX:
 - For use with AIX 4.3.3: Release 4.3.3.0 PCI Cryptographic Coprocessor device driver, file **devices.pci.14109f00.rte-43x**.
 - For use with AIX 5.1: Release 5.1.0.0 PCI Cryptographic Coprocessor device driver, file **devices.pci.14109f00.rte-51c**.

How to Install the Support Program Base Release 2.4.1.0

To install the support program:

1. Log on as **root**.
2. Enter the command **smitty cfgmgr**; press Enter. You are prompted to enter the location of the software to be loaded.
3. Enter the location of the install images you obtained using the procedure described in “How to Download the Software” on page 2-3; press Enter. The software is installed.
4. Exit from **smitty** using the F10 key.

5. To confirm successful installation of the driver, enter the command **lsdev -C -l crypt0**; the system message should reflect status Available. Repeat this command for each CCA Coprocessor installed, changing **crypt0** to **crypt1**, **crypt2**, and so forth as needed.
6. Enter the command **smitty install_latest**.
7. Enter the location of the install images you obtained using the procedure described in “How to Download the Software” on page 2-3; press Enter. The software is installed.
8. When requested, enter **csuf.cca** as the package name for the software to install; press Enter. Press Enter again to continue when prompted ARE YOU SURE.
9. Exit from **smitty** using the F10 key.
10. Read or print `/usr/lpp/csuf/README`; this file contains the latest information about the support program product.
11. Use the configuration utilities to configure the software. Those utilities are described in the next section, “How to Configure the Support Program.”

How to Configure the Support Program

The following utilities and system command are available to configure the software. For more detail, refer to the AIX **man** page for each item.

csufadmin Specifies the system-access permissions associated with the `csufkeys`, `csufappl`, `csufclu` (Coprocessor Load Utility), `csufcnm` (Cryptographic Node Management), and `csufcni` (Cryptographic Node Initialization) utilities.

Default permissions restrict use of these utilities to the root user and to users in the system group. Use the **csufadmin** utility to modify these permissions.

csufappl Specifies the system-access permissions associated with the CCA libraries.

The default permissions restrict use of the CCA libraries to the root user and members of the system group. Use the **csufappl** utility to permit other groups to use the services furnished by the CCA API.

csufkeys Creates and identifies the file and directory names of the locations wherein the cryptographic keys and key lists are stored. The install program defines, in the AIX object data manager (ODM), the following default directories:

```
DES key-storage file:           /usr/lpp/csuf/csufkeys/des.keys
PKA key-storage file:           /usr/lpp/csuf/csufkeys/pka.keys
DES key-record-list directory: /usr/lpp/csuf/csufkeys/deslist
PKA key-record-list directory: /usr/lpp/csuf/csufkeys/pkalist
```

Use the **csufkeys** utility to change the storage locations.

Note: When you initialize key storage using the Cryptographic Node Management utility, ensure that you specify the ODM directories defined by this utility; see “How to Create or Initialize Key Storage” on page 5-21.

odmget Verifies key-storage file names with the odmget system command. You can verify the key-storage names used by the CCA Support Program by entering the following command:

```
odmget csufodm
```

The four parmname attributes specify the following four values:

- csudesds - The file containing the DES key-records
- csupkads - The file containing the PKA key-records
- csudesld - The directory containing the DES key-record-list files
- csupkald - The directory containing the PKA key-record-list files.

When initializing CCA key-storage with either the CNM utility or with the CSNBKSI CCA verb, you must use the file names returned from the ODM. Use the CSUFKEYS utility to change these file names.

The DES_Key_Record_List verb and PKA_Key_Record_List verb produce list files in the /usr/lpp/csuf/csufkeys/deslist and /usr/lpp/csuf/csufkeys/pkalist directories, respectively. These are the default directory names. Depending on your installation, these directory names may have been changed from their default names. These list files are created under the ownership of the environment of the user that requests the list service. Make sure the files created keep the same group ID as your installation requires. This can also be achieved by setting the "set-group-id-on-execution" bit on in these two directories. See the g+s flags in the chmod command for full details. Not doing this may cause errors to be returned on key-record-list verbs.

To assign a default CCA Coprocessor, use the EXPORT command to set the environment variable CSU_DEFAULT_ADAPTER to CRP0n, where n = 1, 2, ..., or 8, depending on which installed CCA Coprocessor you want as the default. If this environment variable is not set when the first CCA verb of a process is called, the CCA software sets Coprocessor CRP01 as the default. If this environment variable is set to an invalid value, you will get an error until the environment variable is set to a valid value.

1 Beginning with Release 2, the CCA implementation provided caching of key records
1 obtained from key storage within the CCA host code. However, the host cache is
1 unique for each host process. If different host processes access the same key
1 record, an update to a key record caused in one process will not affect the contents
1 of the key cache held for other process(es). Beginning with Release 2.41, caching
1 of key records within the key storage system can be suppressed so that all
1 processes will access the most current key records. To suppress caching of key
1 records, use the EXPORT command to set the environment variable CSUCACHE
1 to NO. If this environment variable is not set, or is set to anything other than NO
1 (case is ignored), caching of key records will not be suppressed.

CCA Support Program and AIX File Permissions

The CCA support program relies on file permissions at the "group" level to function correctly. This means that the users and administrators of the CCA support program must have the correct group file permissions on the CCA shared libraries, utilities, and key-storage files and directories in order to be fully functional and run without errors. The csufadmin, csufappl, and csufkeys utilities are provided to aid in this task during installation, but other issues can arise after installation, especially with the key-storage files and directories.

Note: “Key-storage files and directories” are defined as those files and directories contained in the key-storage directory including the top level key-storage directory, that is, in the default configuration, all the files and directories below the /usr/lpp/csuf/csufkeys directory, and the /usr/lpp/csuf/csufkeys directory itself.

For proper operation, the key-storage files and directories *must* have a group ID of the application user group; that is, the “groupname” parameter used when the csufappl utility was run.

Also, as a general rule, all key-storage directories should have file permissions of 770 (drwxrwx---) and be "owned" by root. All key-storage files should have file permissions of 660 (-rw-rw----).

Updating from a Prior Release

When updating the CCA software in an AIX system, be aware that the installation process may delete your host-system key storage files. Make a backup copy of these files prior to starting the installation. Restore these files following the Support Program installation provided that you have not changed master keys in the Coprocessor during the installation process. Note that use of a CLU file¹ of the form CNWxxxxx.CLU (as opposed to using a file such as CEXxxxxx.CLU) will zeroize any preexisting master keys and therefore you should not restore the key-storage files.

How to Review pSeries (RS/6000) Coprocessor Hardware Errors

Errors occurring in the Coprocessor hardware are recorded in the AIX error log. To process and view the log, enter the command

```
errpt -a -N cryptn,libscc.a | more
```

where n is 0, 1, 2, 3, 4, 5, 6, or 7 (for example, crypt0), depending on which CCA Coprocessor log you wish to view.

b
b
b
b
b
b
b
b
b
b
b

b ¹ The Coprocessor Load Utility is explained in Chapter 4, “Loading Software into the Coprocessor” on page 4-1 page=no..

How to Remove the Support Program

If your key-storage files are located in the default directories, back up or save them before you remove the support program; removing the software deletes those key-storage files located in the default directories. For a list of the default directories, see "How to Configure the Support Program" on page 3-3.

To remove the support program:

1. Logon as **root**.
2. Enter the command **rmdev -dl crypt0**; the Coprocessor device driver and related information are removed. Repeat this command for each CCA Coprocessor you plan to remove or relocate, changing **crypt0** to **crypt1**, **crypt2**, and so forth as needed.
3. Enter the command **smitty install_remove**; you are prompted to enter the product names.
4. Enter the product names **csuf.cca**, **csuf.com**, and **devices.pci.14109f00.rte**.
5. Verify that the "REMOVE dependent software" value is **NO**. Also, verify that the "Preview Only" value is **NO**.
6. Press the Enter key.

b How to Install and Remove the Support Program for Windows

b This section includes a description of the support program system requirements and procedures necessary to install and remove the software for Windows NT, Windows 2000, and Windows 2000 Advanced Server. Before you install the 4758 PCI Cryptographic Coprocessor, make sure you have the support program installed. If you accidentally power on a Windows 2000 system with a 4758 PCI Cryptographic Coprocessor installed that does not have the support program installed, you will need to uninstall the "Coprocessor" entry in the hardware device list before the support program will work properly. To do this, follow these steps:

1. From the Windows 2000 Control Panel, open the **Add/Remove Hardware** folder and activate the **Add/Remove Hardware Wizard** to uninstall the "Coprocessor" entry in the hardware device list.
2. If you have multiple Coprocessors installed on the system, repeat step 1 until all of the "Coprocessor" entries in the hardware device list are removed.
3. Restart Windows 2000 with the Support Program already installed.

Make sure your system meets the following requirements:

Hardware:

An IBM-compatible PC with an IBM 4758 PCI Cryptographic Coprocessor installed. For installation instructions regarding the Coprocessor hardware, refer to the *IBM 4758 PCI Cryptographic Coprocessor Installation Manual*.

"What to Do When Migrating from Windows NT to Windows 2000" on page 3-11 explains steps you must perform when migrating from Windows NT 4.0 to Windows 2000.

If you migrate from Windows NT to Windows 2000 (as opposed to performing a new install of Windows 2000) and you are replacing an existing release of the support program with release 2.41, follow the installation procedure described in "What to Do When Migrating from Windows NT to Windows 2000" on page 3-11. You should also use this procedure if you are not sure which installation method was used to install Windows 2000 prior to installing release 2.41 of the support program.

Important: The installation process modifies the system registry; it must be performed by a user with the administrator privilege.

Windows NT and Windows 2000 Requirements

Before you install the support program, make sure your system meets the following requirements:

Hardware:

An IBM-compatible PC with an IBM 4758 PCI Cryptographic Coprocessor installed. During installation of the software, the driver interacts with the Coprocessor to arbitrate interrupt settings, DMA channels, and other system resources. For installation instructions regarding the Coprocessor hardware, refer to the *IBM 4758 PCI Cryptographic Coprocessor Installation Manual*.

Software:

Windows NT Version 4.0 or Windows 2000.

Java runtime environment 1.1.6 through 1.3 (it is possible that a later release of Java will work, but it has not been tested by IBM), available from <http://java.sun.com/j2se>. This is required to use the Cryptographic Node Management utility.

1 Release 2.41 CCA Support Program for Windows NT/Windows 2000.

How to Install the Support Program

To install the support program:

- 1
1. Enter the command **4758241w.exe**.
 2. Follow the online directions.

You have the choice of a typical or custom installation. With a custom installation, you can choose which of these components you wish installed: shared DLLs, samples, device driver, CLU utility, and/or CNM utility.

When prompted to choose the directory location of the software and of the key-storage files used by the Cryptographic Node Management utility and other CCA applications, you can accept the defaults or choose your own directory locations.

3. Adjust the Windows NT/Windows 2000 System Time.

You must set the Windows NT "TZ," time zone, environment variable. The CCA access-control-system logon function requires that the system clock and the Coprocessor clock-calendar be in close synchronization. The CCA support program presumes that the system clock and the time zone settings have been correctly established.

You issue a console command to temporarily set the TZ variable. For example, for the Eastern time zone in the U.S.A.:

b **SET TZ=EST5EDT,4,1,0,7200,10,-1,0,7200,3600**

For proper operation, you must completely set the TZ environment variable.

SET TZ=SSSh[:m[:s]]DDD,sm,sw,sd,st,em,ew,ed,et,shift

Variable	Description	Default Value
SSS	Standard-timezone identifier. It must be three characters, must begin with a letter, and can contain spaces. Zone names are determined by local or country convention. For example, EST stands for Eastern Standard Time and applies to parts of North America.	(none)
h, m, s	The variable h specifies the difference (in hours) between the standard time zone and Coordinated Universal Time (CUT), formerly Greenwich mean time (GMT). You can optionally use m to specify minutes after the hour, and s to specify seconds after the minute. A positive number denotes time zones west of the Greenwich meridian; a negative number denotes time zones east of the Greenwich meridian. The number must be an integer value.	(none)
DDD	Daylight Savings Time (DST) zone identifier. It must be three characters, must begin with a letter, and can contain spaces.	(none)
sm	Starting month (1 to 12) of DST.	0
sw	Starting week (-4 to 4) of DST. Use negative numbers to count back from the last week of the month (-1) and positive numbers to count from the first week (1).	0
sd	Starting day of DST. 0 to 6 if sw != 0 1 to 31 if sw = 0	0
st	Starting time (in seconds) of DST.	0
em	Ending month (1 to 12) of DST.	0
ew	Ending week (-4 to 4) of DST. Use negative numbers to count back from the last week of the month (-1) and positive numbers to count from the first week (1).	0
ed	Ending day of DST. 0 to 6 if ew != 0 1 to 31 if ew = 0	0
et	Ending time of DST (in seconds).	0
shift	Amount of time change (in seconds).	0

To make the Windows NT TZ setting automatic, go to the Windows NT Control Panel, open the **System** folder, then select the **Environment** tab. In the box labeled **Variable**, enter **TZ**, and in the box labeled **Value**, enter the *TZ parameters* as defined for the SET TZ statement. Activate the **OK** button when done.

To make the Windows 2000 TZ setting automatic, go to the Windows 2000 Control Panel, open the **System** folder, then select the **Advanced** tab. Activate the **Environment Variables...** button followed by the **New...** button. In the box labeled **Variable Name**, enter **TZ**, and in the box labeled **Variable Value**, enter the *TZ parameters* as defined for the SET TZ statement. Activate the **OK** button when done.

4. The device driver invokes Coprocessor "PCI-bus chip-set mismatch logic." Netfinity 5000 machines, and possibly some machines from other manufacturers, are incompatible with the chip-set mismatch logic. If you are

using a Netfinity 5000, or if you encounter a hung system within 15 minutes of installing the Coprocessor, you should deactivate the chip-set mismatch logic.

To deactivate the chip-set mismatch logic, use Windows NT/Windows 2000 Explorer and double-click the IdSelect1.reg file (Windows NT) or the idselw21.reg file (Windows 2000) found in the c:\program files\IBM\4758 directory.²

How to Configure the Support Program

To assign a default CCA Coprocessor, use the SET command to set the environment variable CSU_DEFAULT_ADAPTER to CRP0n, where n = 1, 2, ..., or 8, depending on which installed CCA Coprocessor you want as the default. If this environment variable is not set value when the first CCA verb of a process is called, the CCA software sets Coprocessor CRP01 as the default. If this environment variable is set to an invalid value, you will get an error until the environment variable is set to a valid value.

1 Beginning with Release 2, the CCA implementation provided caching of key records
1 obtained from key storage within the CCA host code. However, the host cache is
1 unique for each host process. If different host processes access the same key
1 record, an update to a key record caused in one process will not affect the contents
1 of the key cache held for other process(es). Beginning with Release 2.41, caching
1 of key records within the key storage system can be suppressed so that all
1 processes will access the most current key records. To suppress caching of key
1 records, use the SET command to set the environment variable CSUCACHE to
1 NO. If this environment variable is not set, or is set to anything other than NO
1 (case is ignored), caching of key records will not be suppressed.

How to Remove the Support Program

To remove the support program in Windows NT:

1. Go to the Windows NT Control Panel.
2. Open the **Add/Remove Programs** folder; **IBM 4758 PCI Cryptographic Coprocessor** is displayed in the list of software.
3. Highlight **IBM 4758 PCI Cryptographic Coprocessor**.
4. Activate the **Add/Remove...** button; you are prompted to confirm file deletion.
5. Activate the **Yes** button; the software is removed.

To remove the support program in Windows 2000:

1. Go to the Windows 2000 Control Panel.
2. Open the **Add/Remove Programs** folder; **IBM 4758 PCI Cryptographic Coprocessor** is displayed in the list of software.
3. Highlight **IBM 4758 PCI Cryptographic Coprocessor**.
4. Activate the **Change/Remove...** button; you are prompted to confirm file deletion.
5. Activate the **Yes** button; the software is removed.

² Drive c: is the normal location for the \program files directory tree; your system can differ. If you subsequently need to reactivate the mismatch logic, you can double-click the IdSelect0.reg file.

What to Do When Migrating from Windows NT to Windows 2000

If the Support Program Release 2.41 was installed under Windows NT, then when you migrate from Windows NT to Windows 2000, you must remove and re-install the support program to ensure the correct program files are copied and all entries in the Registry are properly updated. Follow this procedure:

1. Uninstall the support program following the directions for Windows NT/2000 under "How to Remove the Support Program" on page 3-10.
2. Migrate from Windows NT to Windows 2000 if you have not done so already.
3. From the Windows 2000 Control Panel, open the **Add/Remove Hardware** folder and activate the **Add/Remove Hardware Wizard** to uninstall the "Coprocessor" entry in the hardware device list.
4. If you have multiple Coprocessors installed on the system, repeat step 3 until all of the "Coprocessor" entries in the hardware device list are removed.
5. Now install the Support Program Release 2.40, following the directions under "How to Install the Support Program" on page 3-8.

Chapter 4. Loading Software into the Coprocessor

After installing the support program onto the host computer—as described in Chapter 3, “Installing the Support Program”—use the Coprocessor Load Utility (CLU) to load the Coprocessor operating system and CCA application into the Coprocessor.

If you obtain updates to the support program, use the CLU utility to reload the necessary program segments. You can also load software from other vendors using the CLU utility.

This chapter includes:

- Instructions for using the CLU utility to understand what Coprocessors are installed and their status, and to install and uninstall the software that runs within the Coprocessor
- A reference section describing:
 - The Coprocessor memory segments
 - Validating the Coprocessor status
 - The syntax used to invoke the CLU utility
 - CLU utility return codes.

For a deeper understanding of the code loading controls and the security considerations implemented by the Coprocessor, see the research paper *Building a High-Performance Programmable, Secure Coprocessor* that is available on the IBM 4758 product Web site Library page.

Notes:

1. The file locations referenced in this chapter are the default directory paths.
2. Appendix E, “Device Driver Error Codes” describes error codes returned by the Coprocessor device driver. These are often presented in the form of a hexadecimal number such as X'8040xxxx'. You may encounter some of these error situations, especially when you first use the CLU utility and are less familiar with the product and its procedures.
3. The Coprocessor function-control vector (FCV) is loaded by the Cryptographic Node Management utility described in Chapter 5, “Using the CNM and CNI Utilities to Manage the Cryptographic Node.”

How to Load Coprocessor Software

This section provides the procedures you use in loading software into the Coprocessor. You will need to refer to the README file that accompanies the software distribution you are installing for specific .CLU file names. The README file may also provide additional information that amplifies or modifies these general procedures.

You will be instructed to follow this sequence of steps:

1. At a command prompt, change to the directory with the CLU files
2. Determine the software currently resident within the Coprocessor
3. Change the contents of software segments 1, 2, and 3 as appropriate
4. Validate the final contents of the software segments.

Changing the Default Directory and Running CLU

You will need to locate the directory that contains the Coprocessor code files (*.CLU) and possibly the CLU utility. At a command prompt, change to the directory for the code files. If the CLU utility is not in this directory, ensure that your operating system can locate the CLU utility. On Windows NT/2000 systems platforms, the CLU utility must either be located in the default directory or be included in the path statement. The default Coprocessor code directories are:

AIX	/usr/lpp/csuf/clu
Windows NT/2000	\Program Files\IBM\4758\CLU

On Windows NT/2000 systems you can issue a change directory command that includes a space character by enclosing the parameter in quotation marks, for example:

```
cd "\Program Files\IBM\4758\CLU"
```

To run the CLU utility, you enter the program name at the command prompt, **CSUxCLU** where "x" is different for the three operating systems:

AIX	CSUFCLU
Windows NT/2000	CSUNCLU

You can provide parameters interactively to the CLU utility, or you can include these on the command line input. (Details are provided at "Coprocessor Load Utility Syntax" on page 4-8.) Each time that you use CLU you will need to specify a log file name. This is the first parameter and can be included on the command line. In general, when working with a specific Coprocessor, it is strongly recommended you use the Coprocessor serial number as the log file name. You can obtain the serial number from the label on the bracket at the end of the Coprocessor. By always naming the log file with the serial number, you can keep a complete history of status and code changes for the contents of each Coprocessor.

CLU will append information to two log files. If the log files do not exist, they will be created. One log file contains the same information that is normally displayed on your console. The second log file, to which CLU will assign MRL as the file name extension, contains a "machine-readable log." The MRL file is intended for use with an analysis utility.

Subsequent instructions in this section assume that you use CLU interactively. Change to the directory that contains the Coprocessor code files. Start CLU with the name appropriate to your operating system. Respond to the prompts as requested.

CLU obtains the number of installed Coprocessors from the device driver. If you have more than one installed Coprocessor, CLU will interactively ask you for the "number" of the Coprocessor with which you intend to interact. These numbers ("cop.#") range from 0 to 7. To correlate these numbers to a particular

Coprocessor, use the System Status (“SS”) command to learn the number for each of the installed Coprocessors. (See Figure 4-2 on page 4-10.)

Note: The CLU utility can only operate with a Coprocessor when it can obtain exclusive control of the Coprocessor. If any other application (thread) is running that has performed CCA verb calls, all of the Coprocessors that are loaded with CCA will be “busy” and unusable by CLU.

Important: When trying to use CLU, no applications that use CCA or PKCS #11 should be running.

Determining Coprocessor Software Segment Contents

The Coprocessor has three “segments” called segment 1, segment 2, and segment 3. Each segment:

- Has a status
- Holds software
- Holds a validation public key
- Has an owner identifier (except segment 1).

Segment Content

1	“Miniboot,” contains diagnostics and code loading controls
2	CP/Q++ embedded control program
3	CCA, or another application.

You determine the current content and status of the Coprocessor segments using the **ST** command. Figure 4-1 shows a typical ST response. Information in bold text is discussed next.

```

1
1
=====
1 CSUNCLU V2.41 241test.nt.log ST 0   begun Thu Dec 6 14:49:38 2001
1
1 ***** Command ST started. ---- Thu Dec 6 14:49:38 2001
1
1 *** VPD data; PartNum = 04K9434
1 *** VPD data; EC Num = C75605D
1 *** VPD data; Ser Num = PR-01052
1 *** VPD data; Description = IBM4758
1 *** VPD data; Mfg. Loc. = IBM041
1 *** VPD data; Flags = 2000500010000000
1 *** ROM Status; PIC ver: 1900, ROM ver: 202
1 *** ROM Status; INIT: INITIALIZED
1 *** ROM Status; SEG2: RUNNABLE , OWNER2: 2
1 *** ROM Status; SEG3: RUNNABLE , OWNER3: 2
1 *** Page 1 Certified: YES
1 *** Segment 1 Image: 2.41 POST1, MB1
1 *** Segment 1 Revision: 241
1 *** Segment 1 Hash: 1BDF 675F F8C5 B38D 574D EAB7 4542 4523 F9A9 BF27
1 *** Segment 2 Image: 2.41 CP/Q++                200201081244241000000
1 *** Segment 2 Revision: 241
1 *** Segment 2 Hash: 4DB7 7C4E 4792 E3D5 BC4A 48ED 0F40 DE42 698E BB30
1 *** Segment 3 Image: 2.41 CCA                200201150830241000000
1 *** Segment 3 Revision: 241
1 *** Segment 3 Hash: 8FE6 858B 8357 A632 8F36 91EE 6CA3 2C27 BFC9 6967
1 *** Query Adapter Status successful ***
1 Obtain Status ended successfully!
1 ***** Command ST ended. ---- Thu Dec 6 14:50:54 2001

```

Figure 4-1. Typical CLU Status Response

Item Discussion

Ser Num The serial number of the Coprocessor, for example, 41-00004.

Description A statement that describes the type of Coprocessor in general terms. Auditors should review this and other status information to confirm that an appropriate Coprocessor is in use.

ROM Status The Coprocessor must always be in an INITIALIZED state. If the status is ZEROIZED, the Coprocessor has detected a possible tamper event and is in an unrecoverable, non-functional state. (Unintended “tamper” events can be created by improper handling of the Coprocessor. Only remove the batteries when following the recommended battery changing procedure, maintain the Coprocessor in the safe temperature range, and so forth. See the *IBM 4758 PCI Cryptographic Coprocessor Installation Manual*.)

ROM Status SEG2 / SEG3 Several status conditions for SEGment 2 and SEGment 3 exist, including:

- Unowned: currently not in use, no content
- Runnable: contains code and is in a generally usable state.

Owner identifiers are also shown. The standard CCA Support Program is assigned identifier 02 for both segments 2 and 3. **Any other owner identifier** indicates that the software is not the standard IBM CCA product code. In all cases, be certain that the proper software is loaded in your Coprocessor. Unauthorized or unknown software can represent a security risk to your installation.

Segment 1 Image The name and description of the software content of segment 1. For a factory-fresh Coprocessor, the name will include “Factory.” This image and associated validation key will need to be changed.

For a previously loaded Coprocessor, the segment 1 name will probably include “CCA.” Be sure to observe the revision level.

Segment 2 and 3 Images If these segments have Owned status, observe the image name and the revision level. IBM incorporates “CCA” in the image name to indicate that the image is provided as part of the CCA Support Program. Be sure to observe the revision level.

Changing Software Segment Contents

Generally the software within the Coprocessor must be at the same release level as the CCA software in the hosting system. Do not attempt to mix-and-match different release levels except with specific instructions from IBM.

Start the CLU utility and enter the parameters interactively (see “Changing the Default Directory and Running CLU” on page 4-2).

- Enter the log file name (**#####.LOG**, where ##### is the serial number of the Coprocessor).
- Enter the command, **PL**.
- If there are multiple Coprocessors, enter the Coprocessor number.
- Enter the CLU file name as indicated in the README file.

Repeat as required so that the proper software is loaded for segments 1, 2, and 3.

Validating the Coprocessor Segment Contents

After you have loaded or replaced the code in segments 1, 2, and 3, use the CLU VA command to confirm the segment contents and validate the digital signature on the response created by the Coprocessor. Depending on the IBM 4758 Coprocessor in use,¹ issue this command and substitute the file name from the table below for the class-key file name. Note that the file name is the Coprocessor part number followed by "V.CLU."

CSUxCLU #####.LOG VA [cop.#] xxxxxxxx.CLU

Table 4-1. Class-Key Files for Use with CLU VA Command

Coprocessor	File Name
Model 002, 5 volt	40H9952V.CLU
Model 002, 3.3 volt	40H9951V.CLU
Model 023, 5 volt	40H9950V.CLU
Model 023, 3.3 volt	44P1607V.CLU

The README file describes the Image Names that you should observe. "[cop.#]" is the optionally required designator for a particular Coprocessor and defaults to zero.

How to Unload Coprocessor Software and Zeroize the CCA Node

When you use CLU to process a file that surrenders ownership of segment 2, both segment 2 and the subordinate segment 3 are cleared: the code is removed, the validating public key for the segment is cleared, the security-relevant data items held within the Coprocessor for the segment are zeroized, the owner identifiers are cleared, and the segment's status is set to "UNOWNED."

Refer to the README file that accompanies the software distribution you are using for the specific .CLU file name used to surrender ownership of segments 2 and 3. The README file may also provide additional information that amplifies or modifies this general procedure.

Perform these actions:

- Change to the directory that contains the CLU files.
- Start the CLU utility, **CSUxCLU**.
- Respond to the prompts and use the serial number of the Coprocessor in the log file name.
- Use the PL command to surrender segment 2 as indicated in the README file for your platform.

¹ You can refer to the IBM 4758 product Web site (<http://www.ibm.com/security/cryptocards>) FAQ section for the procedure to validate Coprocessor integrity. That topic carries the current list of class-key certificate files.

Notes:

1. You can also zeroize CCA without removing the software by using the CCA reinitialize process. See “How to Initialize (Zeroize) the Node” on page 5-7.
2. IBM does not normally make available a file to restore the factory segment 1 validating key to put the Coprocessor into a condition similar to a factory-fresh product. Segment 1 can only be changed a limited number of times before the available Device Key certificate space is exhausted and the Coprocessor is potentially rendered unusable. If you require a capability to restore the segment 1 factory validating key, and are willing to expose your Coprocessor to a possible lock-up condition, you can obtain the required file from IBM by submitting a query via the Support Form on the IBM 4758 product Web site, <http://www.ibm.com/security/cryptocards>.

Coprocessor Load Utility Reference

If you are interested in additional details concerning the Coprocessor code loading process, continue reading this section. Otherwise, continue reading at Chapter 5, “Using the CNM and CNI Utilities to Manage the Cryptographic Node.”

This reference section describes:

- The Coprocessor memory segments into which you load the software
- The way in which the Coprocessor validates software loads
- The syntax used to invoke the CLU utility
- CLU utility return codes.

Coprocessor Memory Segments

Coprocessor memory segments are organized as follows:

Segment 0	Basic code The basic code manages Coprocessor initialization and the hardware component interfaces. This code cannot be changed after the Coprocessor leaves the factory.
Segment 1	Software administration and cryptographic routines Software in this segment: <ul style="list-style-type: none">• Administers the replacement of software already loaded to Segment 1.• Administers the loading of data and software to segments 2 and 3.• Is loaded at the factory, but can be replaced using the CLU utility.
Segment 2	Embedded operating system The Coprocessor support program includes the CP/Q++ operating system; the operating system supports applications loaded into Segment 3. Segment 2 is empty when the Coprocessor is shipped from the factory.
Segment 3	Application software The Coprocessor support program includes a CCA application program that can be installed into Segment 3. The application

functions according to the IBM CCA and performs access control, key management, and cryptographic operations. Segment 3 is empty when the Coprocessor is shipped from the factory.

Validation of Coprocessor Software Loads

When the Coprocessor is shipped from the factory, it has within it the public key needed to validate replacement software for segment one.

Loading code into Coprocessor segment 2 and segment 3 is a two-step process for each segment.

1. First, an “owner identifier” for a segment is sent to the Coprocessor using an Establish Owner command. The owner identifier is only accepted if the digital signature associated with this identifier can be validated by the public key residing with the immediately lower segment. Once established, ownership remains in effect until a Surrender Owner command is processed by the Coprocessor.
2. Second, a “code load” for a segment is sent to the Coprocessor. Two different commands are available.
 - a. Initially use the Load command. Load command data includes a public-key certificate that must be validated by the public key already residing with the next-lower segment. If the certificate is validated, and if the owner identifier in the Load command data matches the current ownership held by the Coprocessor for the segment, and if the complete Load command data can be validated by the public key in the just-validated certificate, the Coprocessor will accept the code and retain the validated public-key for the segment.
 - b. If a segment already has a public key, a Reload command can be used to replace the code in a segment. The Coprocessor actions are the same as for a Load command, except that the included certificate must be validated by the public key associated with the target segment rather than the key associated with the next-lower segment².

The CP/Q++ embedded operating system, working with the Coprocessor hardware, can store security-relevant data items (SRDI) on behalf of itself and an application in segment 3. The SRDIs are zeroized upon tamper detection, loading of segment software, or a Surrender Owner of a segment. Note that the SRDIs for a segment are not zeroized when using the Reload command. The CCA application stores the master keys, the function-control vector, the access control tables, and retained RSA private keys as SRDI information associated with segment 3.

IBM signs its own software. Should another vendor intend to supply software for the Coprocessor, that vendor’s Establish Owner command and code-signing public-key-certificate must have been signed by IBM under a suitable contract. These restrictions ensure that:

- Only authorized code can be loaded into the Coprocessor

² In this publication the terms “load” and “reload” are employed. Other documentation may refer to these operations as “emergency burn” (EmBurn), and “regular burn” or “remote burn” (RemBurn), respectively.

- Government restrictions are met relating to the import and export of cryptographic implementations.

Coprocessor Load Utility Syntax

This section details the syntax used to invoke the load utility, and describes each function available in it. Use the utility to:

- Ensure that the Coprocessor(s) is not “busy” by ending any application(s) that might have used a Coprocessor. For example, end all applications that use either or both the CCA and PKCS #11 APIs.
- Obtain the release level and the status of software currently installed in the Coprocessor memory segments
- Confirm the validity of digitally signed messages returned by the Coprocessor
- Load and re-load portions of the Coprocessor software
- Reset the Coprocessor.

To invoke the utility:

1. Log on as required by your operating system.
2. Go to the command line.
3. Change directory to the directory containing the CLU utility files. The default directories are:

AIX	<i>/usr/lpp/csufl/clu</i>
NT	<i>\Program Files\IBM\4758\clu</i>

4. Enter the utility name followed by the parameters described below. The utility names are:

AIX	csufclu
NT	csunclu

If you do not supply the necessary parameters, the utility will prompt you as information is required. Optional parameters are enclosed in brackets. The syntax for the parameters following the utility name is:

```
[ logfile_name cmd [cop.#] [datafile_name] [-q ] ]
```

“[” and “]” enclose optional items.

Example: To obtain the Coprocessor status and save the results to the logfile, enter:

```
csufclu #####.log va datafile_name.clu
```

where:

#####.log	Identifies the logfile name, and it is recommended that ##### should be the serial number of the Coprocessor. It is not mandatory to use the serial number, but it can be of value to retain a history of all software changes made to each specific Coprocessor. The utility appends entries to this ASCII text file as it performs the operations requested. A second “machine readable” log file, with a file name of logfile_name.MRL, is also created. This log file can be processed by a program and contains the binary-encoded responses from the Coprocessor.
------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	For information about the contents of this log file, see Appendix C, “Machine-Readable-Log Contents.”
<i>cmd</i>	A two-letter abbreviation representing the command to be run. See “Coprocessor Load Utility Commands.”
<i>cop.#</i>	Provides the Coprocessor number as established by the device driver. This parameter defaults to zero. Coprocessors are designated to the device driver as numbers 0, 1, ..., 7. You can use the serial number information that you obtain with the status or validate commands and the serial number printed on the end-bracket of the Coprocessor to correlate a particular Coprocessor to the <i>cop.#</i> . The utility supports up to eight Coprocessors per machine.
<i>datafile_name</i>	Identifies the data file (drive, directory, and filename) used for the operation requested. <ul style="list-style-type: none"> • For software loads and re-loads it is the filename of the software image you are loading into the Coprocessor. The CCA Support Program README file provides the <i>datafile_name</i>. • When obtaining Coprocessor status with the VA command, it is the class-key certificate filename used to validate the Coprocessor response. The IBM 4758 product Web site (http://www.ibm.com/security/cryptocards) Frequently Asked Questions (FAQ) area contains a description of the procedure for validating the Coprocessor and its code. This description also contains a list of all of the current class-key certificate file names. You can download any required certificate file from the Web site.
<i>-q</i>	Suppresses (quiets) the CLU program output to the standard output device. The status information is nonetheless appended to the log files.

Coprocessor Load Utility Commands

The Coprocessor load utility supports these commands:

SS: System Status

Obtains the part number, serial number, and a portion of the segment-3 software image name for each of the installed Coprocessors provided that these are not being used by some application such as CCA or PKCS #11. See Figure 4-2 on page 4-10.

ST: Obtain Status

Obtains the status of loaded software and the release level of other components. The status is appended in the log files.

VA: Obtain Status with Validation

Obtains the status of loaded software and the release level of other components. The data is transmitted in a message signed by the Coprocessor device key, and then stored in the utility logfile.

The utility uses its built-in public key to validate the one-or-more class-key certificates contained in *datafile_name*. One of these certificates should validate the public key—or chain of public

keys—obtained from the Coprocessor, and confirm that the Coprocessor has not been tampered.

PL: Package Load³

Processes a series of the commands as directed by the contents of the data file to establish segment ownership and to load or reload segment software.

RS: Reset Coprocessor

Resets the Coprocessor. Generally you will not use this command. The command causes the Coprocessor to perform a power-on reset. You may find this command helpful should the Coprocessor and the host-system software lose synchronization. You should end all host-system software processes that are operating with the Coprocessor prior to issuing this command to enable the complete cryptographic subsystem to get to a reset state.

In general, the utility can be invoked by a script file or a command file. When creating a script file or a command file to invoke the utility on an unattended system, add “quiet” syntax **-q** (or **-Q**, **/q**, or **/Q**) to request that nothing be output to the display. By default, the utility returns prompts and messages to the display.

```
1 CSUNCLU V2.41 rrm.log ss begun Thu Dec 6 14:18:45 2001
1 ***** Command ss started. ---- Thu Dec 6 14:18:45 2001
1
1 Card # P/N S/N Segment 3 Description
1 -----
1 0 04K9127 41-00008 CCA 2.41 Segment-3
1 1 04K9127 41-00007 PKCS #11 Application 200011141500
1 2 04K9434 PR-01337 CCA 2.41 Segment-3
1 3 04K9127 41-F0061 CCA 2.41 Segment-3
1 4 04K9041 PR-01200 CCA 2.41 Segment-3
1 5 04K9434 PR-01331 PKCS #11 Application 200011141500
1 6 04K9132 41-00164 CCA 2.41 Segment-3
1 7 04K9434 PR-01052 CCA 2.41 Segment-3
1 *** Query System Status successful ***
1 System Status ended successfully!
1 ***** Command ss ended. ---- Thu Dec 6 14:21:42 2001
1
1 ***** Command ss exited. ---- Thu Dec 6 14:22:42 2001
1
```

Figure 4-2. Typical CLU System Status Response

³ The CLU utility packaged with the CCA Support Program Version 1 used additional commands to control ownership and code loading into the Coprocessor: commands R1, E2, L2, R2, S2, E3, L3, R3, and S3. With this release, these commands are inferred from information contained in the data files that you use with the PL command. A single “PL” file can incorporate information for multiple ownership and loading commands.

Coprocessor Load Utility Return Codes

When the utility finishes processing, it returns a value able to be tested in a script file or in a command file. The returned values and their meanings are:

- 0** OK.
- 1** Command line parameters not valid.
- 2** Cannot access the Coprocessor. Be sure that the Coprocessor and its driver have been properly installed.
- 3** Check the utility logfile for an abnormal condition report.
- 4** No Coprocessor installed. Be sure that the Coprocessor and its driver have been properly installed.
- 5** Invalid Coprocessor number specified.
- 6** A data file is required with this command.
- 7** The data file specified with this command is incorrect or invalid.

Chapter 5. Using the CNM and CNI Utilities to Manage the Cryptographic Node

A computer that provides cryptographic services, such as key generation and digital signature support, is defined here as a *cryptographic node*. The Cryptographic Node Management (CNM) utility and the Cryptographic Node Initialization (CNI) utility provided with the Support Program are tools to set up and manage the CCA cryptographic services provided by a node.

This chapter includes:

- Overview: What the utilities are and how to start them
- How to use the utilities: Three sample scenarios you should consider.

And several sections with details on specific utility topics:

- How to use the CNM utility administrative functions: Things that you should be aware of in the Cryptographic Node Management utility. You should review this material after working through the topic “How to Establish a Test Node” on page 5-3.
- How to create and manage access-control data: Some details about the access-control portion of the Cryptographic Node Management utility.
- How to manage cryptographic keys: Some of the key management things you can accomplish with the Cryptographic Node Management utility.
- Using the CNI utility to establish other nodes: How you can automate use of the Cryptographic Node Management utility using encapsulated procedures.

Note: This chapter describes the major functions of the Cryptographic Node Management utility. For additional information about specific panels and fields, refer to the online help panels included with the utility.

These utilities are written in Java** and require use of a Java runtime environment (JRE). You can also use the Java Development Kit (JDK). For a description of the system setup required to run these utilities, see:

“AIX Requirements” on page 3-2

“Windows NT and Windows 2000 Requirements” on page 3-7.

Overview

Typical users of the Cryptographic Node Management utility and the Cryptographic Node Initialization utility are security administration personnel, application developers, system administrators, and, in some cases, production-mode operators.

Notes:

1. The Cryptographic Node Management utility furnishes a limited set of the CCA API services. After becoming familiar with the utility, you can determine whether it meets your needs or whether you require a custom application to achieve more comprehensive administrative control and key management.
2. Files that you create through use of the CNM utility may be dependent on the release of the Java runtime environment. If you change the release of the Java

runtime environment that you use, files that you have created with the CNM utility might not function correctly with the new release.

3. Files that you create through use of the CNM utility do not operate with the Java runtime environment on other operating system platforms. You must create the CNM-produced files that you use on a machine with the same operating system, and generally with the same release of the Java runtime environment.
4. The CNM utility has been designed for use with a mouse. Use the mouse click instead of the Enter key for consistent results.
5. No help panels are provided for the Master Key Cloning portion of the utility. See "How to Clone a Master Key" on page 5-18.
6. These utilities use the IBM Common Cryptographic Architecture (CCA) API to request services from the Coprocessor. The *IBM 4758 CCA Basic Services Reference and Guide* contains a comprehensive list of the verbs (also known as "callable services" or "procedure calls") provided by the CCA API. You will need to refer to this book and the individual services described herein to understand which commands may require authorization in the various roles that you will define using the procedures described in this chapter.

Cryptographic Node Management Utility Overview

The Cryptographic Node Management (CNM) utility is a Java application that provides a graphical user interface to use in the setup and configuration of IBM 4758 CCA cryptographic nodes. The utility functions primarily to set up a node, create and manage access-control data, and manage the CCA master-keys necessary to administer a cryptographic node.

You can load data objects directly into the Coprocessor or save them to disk. The data objects are usable at other IBM 4758 CCA nodes that use the same operating system and a compatible level of Java.

How to Start the Cryptographic Node Management Utility: To start the CNM utility:

- On AIX systems, enter **csufcnm** on the command line.
- On Windows NT/2000 systems:
 - Change directory to **\program files\ibm\4758\cnm**
 - Enter **csuncnm** on the command line.

The CNM utility logo and then the main panel are displayed.

Cryptographic Node Initialization Utility Overview

The Cryptographic Node Initialization (CNI) utility runs scripts that you create using the *CNI Editor* within the Cryptographic Node Management utility. These scripts are known as *CNI lists*. The CNI utility can run the Cryptographic Node Management utility functions necessary to set up a node; for example, it can be used to load access-control roles and profiles.

As you create a CNI list, you specify the disk location of the data objects that the Cryptographic Node Initialization utility will load into the target nodes. After creating a CNI list, you can distribute the CNI list and any accompanying data files (for roles, profiles, and so forth) to nodes where the CNI utility will be used for an

“automated” setup. The source node and all nodes running the distributed CNI list must employ the same operating system and a compatible level of Java.

The Cryptographic Node Initialization utility is further explained in “Using the CNI Utility to Establish Other Nodes” on page 5-24.

How to Use the Utilities, Sample Scenarios

The following scenarios illustrate how to use the utilities:

1. Establish a test node to be used to develop applications or establish procedures for using the Cryptographic Node Management utility. *First-time users should follow this procedure to begin experimentation with the utility and the Coprocessor.*
2. Establish nodes for a production environment using key parts. This scenario employs CNI lists to automate establishment of “target” production nodes.
3. Clone a master key from one Coprocessor to another Coprocessor. This is a procedure of interest to very high-security installations that employ multiple Coprocessors.

The purpose of the scenarios is to illustrate how the procedures described in this chapter can be used. Where appropriate, a scenario cross-refers to sections with more detailed information.

If you are not familiar with the Coprocessor's CCA access-control system, see “Access-Control Overview” on page 5-10 and “Initial State of the Access-Control System” on page 5-10. Here you will find an explanation of terms like *role*, *initial-DEFAULT role*, and *user profile*. The scenarios assume that the access-control system is in its initial state.

Note: These scenarios are instructional only. You are encouraged to determine the procedures best suited for your specific environment. Be sure to review the contents of Chapter 6, “Observations on Secure Operations.”

How to Establish a Test Node

In this scenario, a single developer sets up a node to allow unlimited access to cryptographic services.

Important: The resulting cryptographic node should not be considered secure because under this scenario many sensitive commands are permitted unrestricted use.

1. Install the Coprocessor and the CCA Cryptographic Coprocessor Support Program as described in the previous chapters. Start the Cryptographic Node Management utility as described at “How to Start the Cryptographic Node Management Utility” on page 5-2.

Remember that you must have installed an appropriate level of the Java Runtime Environment (JRE) or the Java Development Kit (JDK).

2. If you have more than one Coprocessor with CCA installed, specify to the CNM utility which Coprocessor you want to use. From the **Crypto Node** pull-down menu, select **Select Adapter**. You will see a drop-down list of available adapter numbers (ranging from one up to a maximum of eight). Choose an

- adapter (Coprocessor) from the list. If you do not use the Select Adapter pull-down to choose an adapter, the default adapter (Coprocessor) is used.
3. Synchronize the clock within the Coprocessor and host computer. From the **Crypto Node** pull-down menu, select **Time**; a sub-menu is displayed. From the sub-menu, select **Set**; the clocks are synchronized.
 4. Use the CNM utility to permit all commands in the DEFAULT role. From the **Access Control** pull-down menu, select **Roles**. Highlight the **DEFAULT** entry and select **Edit**. You will see a screen that shows which commands are already enabled and which commands are not enabled by the DEFAULT role. Select **Permit All**. Then load the modified role back into the Coprocessor by selecting **Load** and then **OK**.

Before selecting Cancel, you could have saved a copy of this “all-commands-enabled” role to your file system using the **Save** button and assigning a file name. You must also select the folder (directory, library) where you will save the role.

For more detail, see “How to Define a Role” on page 5-11.

Finish this task by selecting **Cancel**.

5. Load the function-control vector into the Coprocessor. From the **Crypto Node** pull-down menu, select **Authorization**; a sub-menu is displayed. From the sub-menu, select **Load** to specify and load the function-control vector.

The FCV file that you need to specify is the one that you downloaded from the Web. FCVs usually have file names such as “CCA5203.FCV” and can be found by using the file search utility available with your operating system.

6. Install a master key. From the **Master Key** pull-down menu, select **Auto Set...**; you are prompted to verify the command. Select **Yes**; the Coprocessor generates and sets a master key.

The master key installed with Auto Set has actually passed through the main memory of your system processor as key parts. For production purposes, you should use a more secure method of establishing a master key such as random generation or installation of known key-parts entered by two or more individuals. These options are also accessed from the Master Key pull-down menu.

For more detail, see “How to Auto-Set or Randomly Generate the Master Key” on page 5-17.

7. *Key storage* is a CCA term that describes a place where the support program can store DES and RSA cryptographic keys under names that you (or your applications) define. If you will use key storage, one or both of the DES and the RSA (“PKA”) key-storage files must be initialized. See “How to Create or Initialize Key Storage” on page 5-21.

How to Establish Nodes in a Production Environment

In this scenario, the responsibility for establishing cryptographic nodes is divided among three individuals, namely, an access-control administrator and two key-management officers. The administrator sets up the node and its access-control system, then the key-management officers load a master key and any required key-encrypting key(s). The key-encrypting keys can be used as transport keys to convey other keys between nodes.

Note that this scenario is focused on installing master keys and high-level, inter-node DES key-encrypting keys from *key parts*. The CCA implementation supports alternatives such as random master-key generation and distribution of DES keys using techniques based on RSA public-key technology. The key-part technique assumes that there are two *key-management officers* who can be trusted to perform their tasks and to not share their key-part information. This implements a *split knowledge* policy. The access-control system is set up to enforce *dual control* by separating the tasks of the first and second officers.

In this scenario, the access-control administrator uses the Cryptographic Node Management utility to prepare CNI lists for the target node(s). The CNI lists automate the process of using the Cryptographic Node Management utility at the target node. The administrator prepares a CNI list for the tasks performed by the target node access-control administrator and the two key-management officers. The administrator must know what commands require authorization in the target node under different conditions, including:

- Normal, limited operation (when the default role is used)
- When performing the access-control-administrator tasks
- When performing each of the key-management-officer tasks
- And under any other special circumstances using additional roles and profiles.

The administrator authorizes commands in the various roles to ensure that only those commands actually required are enabled. Sensitive commands, such as loading a first key part or loading subsequent key part(s), are only enabled in roles for users with the responsibility and authority to utilize those commands. It is important to separate the responsibilities so that policies such as “split knowledge” and “dual control” are enforceable by the Coprocessor’s access-control system.

For more detail, see “How to Create and Manage Access-Control Data” on page 5-9.

Access-Control-Administrator Procedure

In this task, the access-control administrator uses the Cryptographic Node Management utility to prepare CNI lists for the target node(s). To set up the node and create its access-control data, the access-control administrator can:

1. On an established node, start the Cryptographic Node Management utility
2. Create and save to disk the access-control data for the target node, including:
 - Supervisory roles and user profiles for the access-control administrator and the key-management officers
 - A DEFAULT role to replace the initial-DEFAULT role.

For more detail, see “How to Create and Manage Access-Control Data” on page 5-9. For information about creating a CNI list, see “Using the CNI Utility to Establish Other Nodes” on page 5-24.

- a. Create a Cryptographic Node Initialization list to:
 - 1) Synchronize the clock-calendar within the Coprocessor and host computer
 - 2) Load the access-control data
 - 3) Log on as an access-control administrator
 - 4) Load the replacement DEFAULT role
 - 5) Load the function-control vector
 - 6) Log off.

- b. Create a CNI list for the first key-management officer:
 - 1) Log on for the first key-management officer
 - 2) Load a first master-key key-part
 - 3) As required, load first-part key-encrypting-key information
 - 4) Log off.
 - c. Create a CNI list for the second key-management officer:
 - 1) Log on for the second key-management officer
 - 2) Load a second master-key key-part
 - 3) As required, load second-part key-encrypting-key information
 - 4) Log off.
3. Install the Coprocessor and the support program onto the target node(s).

Note to AIX Users: By default, use of support program utilities is restricted to the root user and the *system* group. See “How to Configure the Support Program” on page 3-3 for information about setting the permissions associated with the utilities.
 4. Transport to the target nodes the access-control data and the function-control vector specified in the CNI list.
 5. With the involvement of the key-management officers, on each target node run the CNI lists developed in steps 2a, b, and c. See “Using the CNI Utility to Establish Other Nodes” on page 5-24.

The target nodes are now ready to provide cryptographic service.

Key-Management-Officer Procedures

The key-management officers have two tasks:

- Prepare the key parts for eventual use at the target node(s)
- Load the key parts at the target nodes.

You have to decide how the key parts will be transported from the point of generation to the point of installation. There are several reasonable scenarios:

1. Generate the key parts at a central place and transport these on diskettes
2. Generate the key parts at a central place and transport these on paper forms
3. Generate the key parts at the point and time of (first) installation. If the key parts will be needed at another time, either to reload or to share with another node, then how the key parts will be transported has to be decided.

You should review the specific capabilities of the Cryptographic Node Management utility by working with the utility. Then review the specific approach that you select and test the Cryptographic Node Initialization that has been prepared in conjunction with the access-control administrator.

For more detail, see “How to Manage Cryptographic Keys” on page 5-15.

How to Use the CNM Administrative Functions

This section describes how to use the Cryptographic Node Management utility to:

- Optionally choose among multiple Coprocessors
- Initialize (or “zeroize”) the Coprocessor
- Log on to and off of the Coprocessor
- Load the Coprocessor function-control vector
- Configure the utility defaults
- Synchronize the clock-calendars within the Coprocessor and the host computer
- Poll status information about the Coprocessor and the CCA application.

How to Choose a Coprocessor

If your system has multiple Coprocessors loaded with the CCA code, generally you will need to select the specific Coprocessor with which you wish to operate upon. If you do not make a selection, you will operate with the default Coprocessor. Once you make a Coprocessor selection, that selection remains in effect for the current utility session or until you make a different selection within the utility session.

To select an adapter (Coprocessor) you want to use, from the **Crypto Node** pull-down menu, select **Select Adapter**. You will see a drop-down list of available adapter numbers (ranging from one up to a maximum of eight). Choose an adapter (Coprocessor) from the list. If you do not use the Select Adapter pull-down to choose an adapter, the default adapter (Coprocessor) is used.

Notes:

1. When using the CLU utility, Coprocessors are referenced as 0, 1, ..., 7. Any particular Coprocessor may or may not have the CCA application installed. (For example, some Coprocessors may have the PKCS #11 application installed.) With the CNM utility (and other applications that use the CCA API), the Coprocessors loaded with the CCA application are designated 1, 2, ..., 8. These new identifiers are assigned by CCA as it scans all of the installed Coprocessors for those loaded with the CCA application.
2. When coding a CCA application, keywords CRP01, CRP02, ..., CRP08 are used to “allocate” a Coprocessor. These correspond to the numbers 1, 2, ..., 8 used in the CNM utility pull-down.

How to Initialize (Zeroize) the Node

You can restore the CCA node to its initial state, provided that the role you are operating under (the default role or a logged-on role) permits use of the Initialize Device command (offset X'0111'). Use of this command causes clearing of all:

- master-key registers
- retained and registered keys
- roles and profiles and restoring the access control to its initial state (see “Initial State of the Access-Control System” on page 5-10).

To initialize the CCA node, select **Initialize** from the **Crypto Node** pull-down menu. You will be asked to confirm your intent to perform this major action.

How to Log On and Off the Node

To log on, select **Logon** from the **File** pull-down menu. To log off, select **Logoff** from the **File** pull-down menu.

Note: With the exception of the DEFAULT role, access to the Coprocessor is restricted by passphrase authentication.

How to Load the Function-Control Vector

A function-control vector (FCV) is a signed value provided by IBM to enable the CCA application in the Coprocessor to provide a level of cryptographic service consistent with applicable import and export regulations. Under the current regulations all users are entitled to the same level of cryptographic functionality. Therefore, IBM now supplies a single FCV with the CCA Support Program.

You use the CNM utility to load the function-control vector into the Coprocessor. The FCV file is named "CCA5203.FCV." You can locate this file using the file-name search tool provided with your operating system.

To load the function-control vector:

1. From the **Crypto Node** pull-down menu, select **Authorization**; a sub-menu is displayed.
2. From the sub-menu, select **Load** to specify the function-control vector file on disk; the utility loads the function-control vector.

How to Configure the Cryptographic Node Management Utility

The configuration panel of the CNM utility allows you to indicate directory paths for the files you create with the utility. However, the utility generally *does not* use the paths that you store in the configuration panel. Instead, the default paths are stored in the Windows NT/2000 Registry or in the AIX object data manager (ODM). You may find the configuration panel a useful place to record where you intend to keep the various classes of data items.

How to Synchronize the Clock-Calendars

The Coprocessor uses its clock-calendar to record time and date and to prevent replay attacks in passphrase-based profile authentication. After installing the Coprocessor, synchronize its clock-calendar with that of the host system.

To synchronize the clock-calendars:

1. From the **Crypto Node** pull-down menu, select **Time**; a sub-menu is displayed.
2. From the sub-menu, select **Set**; the clock-calendars are synchronized.
3. Answer **Yes** to synchronize the clock-calendars with the host.
4. Finish this task by selecting **OK**.

How to Obtain Status Information

You can use the Cryptographic Node Management utility to obtain the status of the Coprocessor and the CCA application. The following status panels are available:

- *CCA Application*: Displays the version and the build date of the application. Also displays the status of the master-key registers. For information about these registers, see “How to Manage the Master Key” on page 5-16.
- *Adapter*: Displays the Coprocessor serial number, ID, and hardware level.
- *Command History*: Displays the five most recent commands and subcommands sent to the Coprocessor.
- *Diagnostics*: Indicates whether any of the Coprocessor tamper-sensors have been triggered, whether any errors have been logged, and reflects the status of the Coprocessor batteries. To view the AIX Coprocessor log, see “How to Configure the Support Program” on page 3-3.
- *Export Control*: Displays the maximum strength of the cryptographic keys used by the node, as defined by the function-control vector resident within the Coprocessor.

To view the status panels:

1. From the **Crypto Node** pull-down menu, select **Status**. The CCA Application status is displayed.
2. To select other status information, use the buttons at the bottom. The new panel is displayed.
3. Finish this task by selecting **Cancel**.

How to Create and Manage Access-Control Data

The access-control system of the CCA Cryptographic Coprocessor Support Program defines the circumstances under which the Coprocessor can be used. It does this by restricting the use of CCA commands. For a list of these commands, see Appendix A, “CCA Access-Control Commands.” Also see *Required Commands* at the end of each verb description in the *IBM 4758 CCA Basic Services Reference and Guide*.

An administrator can give users differing authority, so that some users can use CCA services not available to others. This section includes an overview of the access-control system and instructions for managing your access-control data. You need to know which commands are required and under what circumstances. You also need to consider that some commands should be authorized only for selected, trusted individuals, or for certain programs that operate at specific times. Generally, you should only authorize those commands that are required so as not to inadvertently enable a capability that could be used to weaken the security of your installation(s). You will obtain the information about command use from the documentation for the applications that you intend to support. See Chapter 6, “Observations on Secure Operations” for additional guidance on this topic.

Access-Control Overview

The access-control system restricts or permits the use of commands based on roles and user profiles. Use the Cryptographic Node Management utility to create roles that correspond to the needs and privileges of assigned users.

To access the privileges assigned to a role (those that are not authorized in the default role), a user must log on to the Coprocessor using a unique user profile. Each user profile is associated with a role. (Multiple profiles can use the same role.) The Coprocessor authenticates logons using the passphrase associated with the profile that identifies the user.

Note: The term “user” applies to both humans and programs.

The Coprocessor always has at least one role—the DEFAULT role. Use of the DEFAULT role does not require a user profile. Any user can use the services permitted by the DEFAULT role without logging onto or being authenticated by the Coprocessor.

For example, a basic system might include the following roles:

- **Access-Control Administrator:** Can create new user profiles and modify the access rights of current users.
- **Key-Management Officer:** Can change the cryptographic keys. (This responsibility is best shared by two or more individuals making use of rights to enter “first” or “subsequent” key parts.)
- **General User:** Can use cryptographic services to protect his or her work, but has no administrative privileges. If your security plan does not require logon authentication for general users, address their requirements in the DEFAULT role.

Note: Few individuals would be assigned the roles of key-management officer or access-control administrator. Generally, the larger population would not log on and thus would have rights granted in the DEFAULT role.

Initial State of the Access-Control System

After you have loaded the CCA software support into segment 3 of the Coprocessor—or after the access-control system is initialized—no access-control data exists except for an initial-DEFAULT role that allows unauthenticated users to create and load access-control data. For a full description of this role, see Appendix B, “Initial DEFAULT-Role Commands” on page B-1.

After creating the roles and profiles needed for your environment—including the supervisory roles necessary to load access-control data and to manage cryptographic keys—remove all permissions assigned to the DEFAULT role. Then, add only those permissions you want to grant to unauthenticated users.

Important: The cryptographic node and the data it protects are not secure while the DEFAULT role is permitted to load access-control data.

How to Define a Role

A role defines permissions and other characteristics of the users assigned to that role. To define a role:

1. From the **Access Control** pull-down menu, select **Roles**; a list of currently defined roles is displayed.
2. Select **New** to display the Role Definition panel; see Figure 5-1. At any time in the process, select **List** to return to the list of currently defined roles.

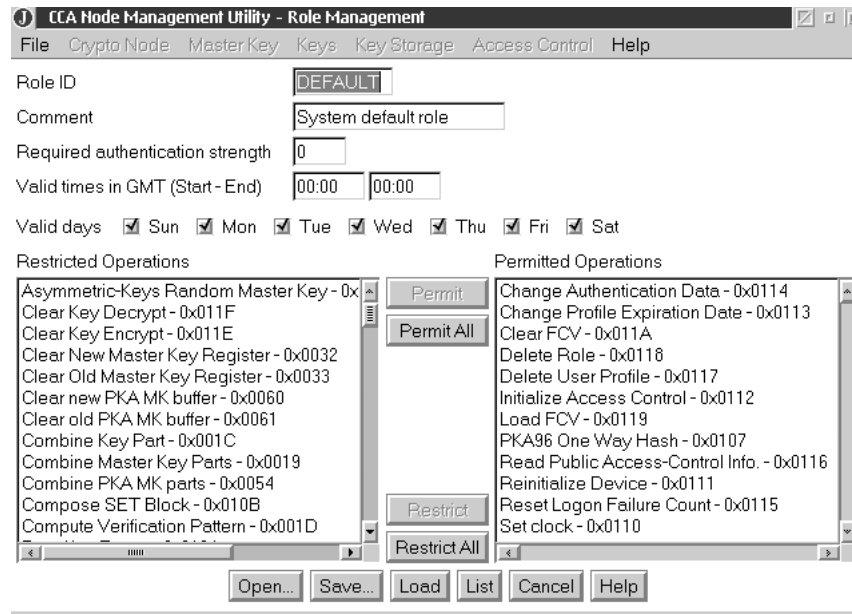


Figure 5-1. The Role Definition Panel

3. Define the role:

Role ID

A character string that defines the name of the role. This name is contained in each user profile associated with this role.

Comment

An optional character string.

Required Authentication Strength

When a user logs on, the strength of the authentication provided is compared to the strength level required for the role. If the authentication strength is less than that required, the user cannot log on. Currently only the passphrase authentication method is supported; use a strength of 50.

Valid Time and Valid Days of the Week

These values determine when the user can log on. Note that these times are Coordinated Universal Time. If you are not already familiar with the access-control system, you may refer to chapter 2 of the *CCA Basic Services Reference and Guide*.

Restricted Operations and Permitted Operations

A list defining the commands the role is allowed to use.

Each CCA API verb requires one or more commands to obtain service from the Coprocessor. The user requesting service must be assigned to a role that permits those commands needed to run the verb.

For more information about CCA verb calls and commands, refer to the *IBM 4758 CCA Basic Services Reference and Guide*. For a list of the commands, and suggestions for their use, see Appendix A, "CCA Access-Control Commands."

4. Select **Save...** to save the role to disk.
5. Select **Load** to load the role into the Coprocessor.

How to Edit Existing Roles

Use the Cryptographic Node Management utility to:

- Edit a disk-stored role
- Edit a Coprocessor-stored role
- Delete a Coprocessor-stored role.

Tip: Any existing role can be used as a template to create a new role. When you open a saved role, the existing information is displayed in the Role Definition panel. You need only modify or enter information specific to the new role, then give it a new Role ID and load or save it.

How to Edit a Disk-Stored Role

To edit a role stored on disk:

1. From the **Access Control** pull-down menu, select **Roles**; a list of currently defined roles is displayed.
2. Select **Open...**; you are prompted to choose a file.
3. Open a file; data is displayed in the Role Definition panel.
4. Select **Save...** to save the role to disk; select **Load** to load the role into the Coprocessor.

How to Edit a Coprocessor-Stored Role

To edit a role stored in the Coprocessor:

1. From the **Access Control** pull-down menu, select **Roles**; a list of currently defined roles is displayed.
2. Highlight the role you want to edit.
3. Select **Edit**; data is displayed in the Role Definition panel.
4. Edit the role.
5. Select **Save...** to save the role to disk; select **Load** to load the role into the Coprocessor.

How to Delete a Coprocessor-Stored Role

Important: When you delete a role, the Cryptographic Node Management utility does not automatically delete or re-assign the user profiles associated with that role. Be sure to delete or re-assign the user profiles associated with a role *before* you delete the role.

To delete a role stored in the Coprocessor:

1. From the **Access Control** pull-down menu, select **Roles**; a list of currently defined roles is displayed.
2. Highlight the role you want to delete.
3. Select **Delete...**; the role is deleted.

How to Define a User Profile

A user profile identifies a specific user to the Coprocessor. To define a user profile:

1. From the **Access Control** pull-down menu, select **Profiles**; a list of currently defined profiles is displayed.
2. Select **New** to display the Profile Management panel; see Figure 5-2.

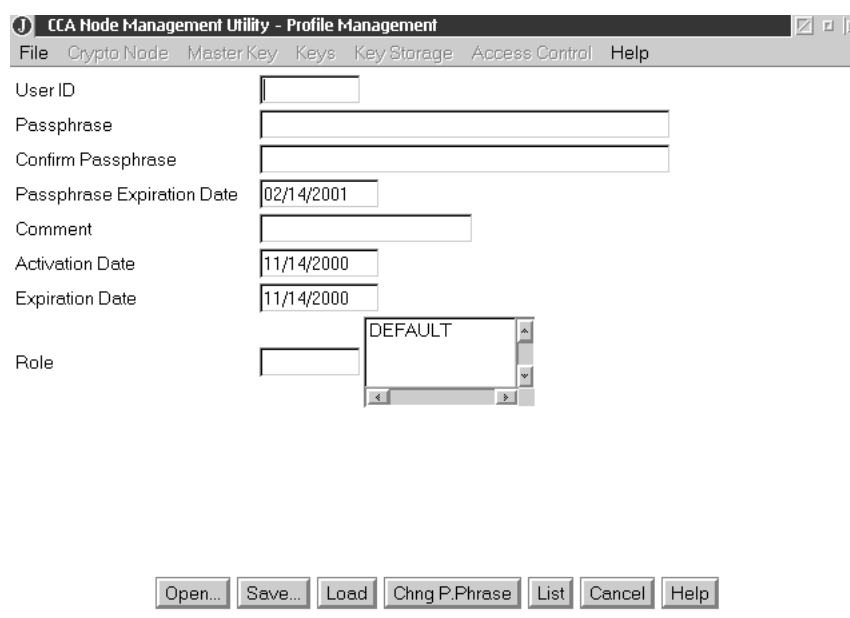


Figure 5-2. The Profile Management Panel

3. Define the user profile:

User ID

The “name” given to a user of the PCI Cryptographic Coprocessor.

Passphrase

The character string that the user must enter to gain access to the cryptographic node.

Passphrase Expiration Date

The expiration date for the passphrase. The utility will set this by default to 90 days from the current date. You can change the expiration date. Every passphrase contains an *expiration date*, which defines the lifetime of that passphrase. This is different from the expiration date of the profile itself.

Comment

An optional character string.

Activation and Expiration Dates

These values determine the first and last dates when the user can log on.

Role

The name of the role that defines the permissions granted to the profile.

4. Select **Save...** to save the profile to disk; select **Load** to load the profile into the Coprocessor.
5. Select **List** to return to the list of currently defined profiles.

How to Edit Existing User Profiles

Use the Cryptographic Node Management utility to:

- Edit a disk-stored user profile
- Edit a Coprocessor-stored user profile
- Delete a Coprocessor-stored user profile
- Reset the user-profile-failure count.

How to Edit a Disk-Stored User Profile

To edit a profile stored on disk:

1. From the **Access Control** pull-down menu, select **Profiles**; a list of currently defined profiles is displayed.
2. Select **Open...**; you are prompted to choose a file.
3. Open a file; data is displayed in the User Profile Definition panel.
4. Edit the profile.
5. Select **Save...** to save the profile to disk; select **Load** to load the profile into the Coprocessor.

How to Edit a Coprocessor-Stored User Profile

To edit a profile stored in the Coprocessor:

1. From the **Access Control** pull-down menu, select **Profiles**; a list of currently defined profiles is displayed.
2. Highlight the profile you want to edit.
3. Select **Edit**; data is displayed in the User Profile Definition panel.
4. Edit the profile.
5. Select **Save...** to save the profile to disk; select **Replace** to load the profile into the Coprocessor.

How to Delete a Coprocessor-Stored User Profile

To delete a profile stored in the Coprocessor:

1. From the **Access Control** pull-down menu, select **Profiles**; a list of currently defined profiles is displayed.
2. Highlight the profile you want to delete.
3. Select **Delete...**; the profile is deleted.

How to Reset the User-Profile-Failure Count

To prevent unauthorized logons, the access-control system maintains a logon-attempt-failure count for each user profile. If the number of failed attempts for a profile exceeds the limit defined in the profile, the offending profile is disabled. To reset the failure count:

1. From the **Access Control** pull-down menu, select **Profiles**; a list of currently defined profiles is displayed.
2. Highlight the profile.
3. Select **Reset FC**; a confirmation dialog box is displayed.
4. Select **Yes** to confirm; the logon-attempt-failure count is set to zero.

How to Initialize the Access-Control System

When you initialize the access-control system, the Cryptographic Node Management utility:

- Clears the access-control data in the Coprocessor
- Furnishes the DEFAULT role with the commands required to load access-control data.

Important: The cryptographic node and the data it protects are not secure while the DEFAULT role is permitted to load access-control data.

Successfully performing this action removes installation-installed access controls and keys and is therefore a very sensitive operation that could render your node inoperable for production. Some installations will choose to remove authorization for this function from their Coprocessor's roles. In this event, if you wish to initialize the CCA cryptographic node you must remove the CCA software from the Coprocessor and re-install the CCA software.

To initialize the access-control system:

1. From the **Access Control** pull-down menu, select **Initialize...**; a confirmation dialog box is displayed.
2. Select **Yes** to confirm; the utility initializes the access-control system.

How to Manage Cryptographic Keys

This section describes how to use the Cryptographic Node Management utility to:

- Manage the master key
- Manage primary key-encrypting keys (KEKs)
- Reset and manage DES and PKA key-storage.

Key types are defined as follows:

The **master key** is a special KEK stored in the clear (not enciphered) and kept within the Coprocessor secure module. It is used to encipher other keys so that those keys can be stored outside of the secure module. The master key is a 168-bit key formed from three 56-bit parts.

Primary key-encrypting keys are DES keys shared by cryptographic nodes and are sometimes referred to as transport keys. They are used to encipher other keys shared by the nodes. Primary keys, like the master key, are

installed from key parts. Knowledge of the key parts can be shared in part by two people to effect a split-knowledge, dual-control security policy.

Other DES keys and PKA keys are enciphered keys used to provide cryptographic services. They include MAC keys and private RSA keys.

Note: When exchanging clear key-parts, ensure that each party understands how the exchanged data is to be used, since the management of key parts varies among different manufacturers and different encryption products.

How to Manage the Master Key

A master key is used to encrypt local-node working keys while they are stored external to the Coprocessor. CCA defines three master-key registers:

- The **current-master-key register** stores the master key currently used by the Coprocessor to encrypt and decrypt local keys
- The **old-master-key register** stores the previous master key and is used to decrypt keys enciphered by that master key
- The **new-master-key register** is an interim location used to store master-key information as accumulated to form a new master key.

The CCA Version 2 Support Program uses two sets of master-key registers, one set for encrypting DES (symmetric) keys, and one set for encrypting public-private (asymmetric) keys.

For information about checking the contents of these registers, see “How to Obtain Status Information” on page 5-9.

Notes:

1. Programs that use the Version 2 CCA API master-key-administration verbs, `Master_Key_Process` and `Master_Key_Distribution`, can use a keyword to steer operations to the asymmetric master-key registers, to the symmetric master-key registers, or both sets of master-key registers. The Cryptographic Node Management utility uses the *both* option. If you use another program to load master keys, and if this program specifically operates on either the symmetric or asymmetric master-key registers, in general you will no longer be able to use the Cryptographic Node Management utility to administer master keys.
2. If your installation has multiple Coprocessors loaded with CCA, you will need to independently administer the master keys in each Coprocessor.

This section describes how to:

- Verify the current master-key
- Load a master key automatically
- Load a new master-key from parts
- Clone a master key.

How to Verify an Existing Master Key

The utility generates a verification number for each master key stored in the master-key registers. This number identifies the key, but does not reveal information about the actual key value.

To view a master-key-verification number:

1. From the **Master Key** pull-down menu, select **Verify**; a sub-menu is displayed.
2. From the sub-menu, select a master-key register; the verification number for the key stored in that register is displayed.

How to Auto-Set or Randomly Generate the Master Key

The Cryptographic Node Management utility can auto-set a master key into the Coprocessor; its key value cannot be viewed from the utility.

Important: If a master key of unknown value is lost, you cannot recover the keys enciphered under it.

To automatically load the master key:

1. From the **Master Key** pull-down menu, select **Auto Set...** or select **Random**; you are prompted to verify the command.
2. Select **Yes**; the Coprocessor generates and sets a master key.

Notes:

1. Use of Random is preferred since the Auto-Set option passes clear key-parts through host-system memory.
2. When you set or auto-set a master key, you must reencipher all keys enciphered under the former key. See “How to Reencipher Stored Keys” on page 5-22.

How to Load a New Master-Key from Key Parts

To set a new master-key into the Coprocessor, load the first, any middle, and last key parts into the new-master-key register, and then set the new master-key. To effect this:

1. From the **Master Key** pull-down menu, select **Parts**; the Load Master Key panel is displayed; see Figure 5-3.

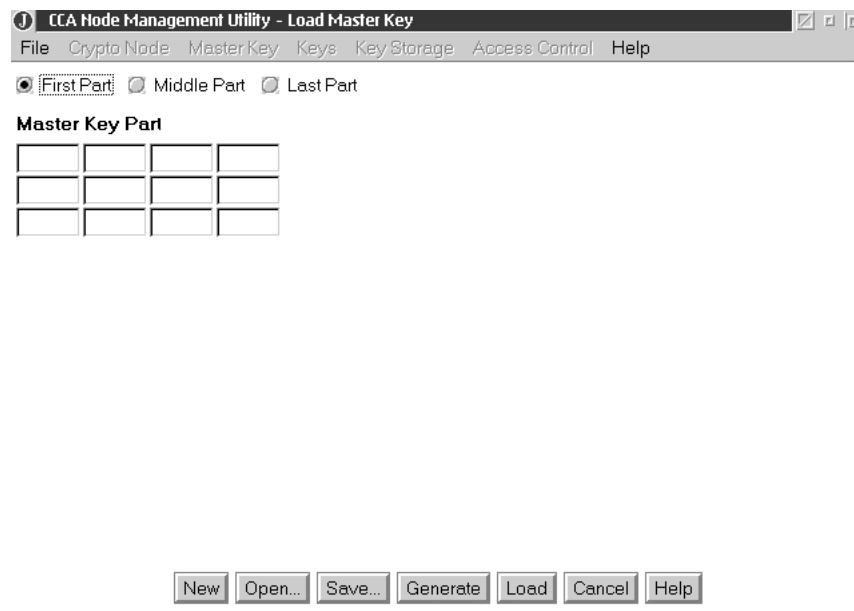


Figure 5-3. The Load Master Key Panel

2. Select the radio button for the key part you are editing (First Part, Middle Part, or Last Part).
3. Enter data by one of the following:
 - Select **New** to clear data entered in error.
 - Select **Open...** to retrieve pre-existing data.
 - Select **Generate** to fill the fields with Coprocessor-generated random numbers.
 - Manually enter data into the “Master Key Part” fields; each field accepts four hexadecimal digits.
4. Select **Load** to load the key part into the new-master-key register; select **Save...** to save the key part to disk.

Important: Key parts saved to disk are not enciphered. Consider storing the key parts on diskettes that are locked in safes.

Note: When you create a key from parts, you must have both a first part and a last part; middle part(s) are optional.
5. Repeating the preceding steps, load into the new-master-key register the remaining key parts.

Note: For split-knowledge security policy, different people must enter the separate key parts. To enforce a dual-control security policy, the access-control system should assign the right to enter a first key part to one role and the right to enter subsequent key part(s) to another role. Then authorized users log on and perform the loading of their respective key part.
6. From the **Master Key** pull-down menu, select **Set...**; the utility:
 - a. Transfers the data in the current-master-key register to the old-master-key register, and deletes the former old-master key.
 - b. Transfers the data in the new-master-key register to the current-master-key register.

After setting a new master-key, reencipher the keys currently in storage. See “How to Reencipher Stored Keys” on page 5-22.

How to Clone a Master Key

This scenario explains the steps involved in *cloning* a master key from one Coprocessor to another Coprocessor. The term cloning is used rather than copying since the master key will be split into *shares* for transport between the Coprocessors. The technique is explained at some length in “Understanding and Managing Master Keys” in Chapter 2 of the *CCA Basic Services Reference and Guide*. Appendix F, “Master-Key Cloning” on page F-1 provides a step-by-step procedure that you can follow. The material in this chapter provides background information that can permit you to vary the procedure.

Cloning of the master key involves two or three nodes:

- The master-key source node
- The master-key target node
- The “share administration” (SA) node.
(The SA node can also be either the source or the target node.)

The Cryptographic Node Management utility can store various data items involved in this process in a “data base” that you can carry on diskette or FTP between the different nodes. One data base is, by default, known as 'sa.db' and contains the information about the SA key and keys that have been certified. The target node where the master key will be cloned also has a data base known by default as the 'csr.db'.

You can accomplish these tasks using the Cryptographic Node Management utility:

1. Set up the nodes in a secure manner with access-control roles and profiles and master keys.

You will need a role and profile(s) at the source and target nodes for each user who will obtain or store share_i, 1 ≤ i ≤ n. Processing of share_i is a separate command so that, if you wish, your roles can insure that independent individuals are involved with obtaining and installing the different shares.

Consider the use of random master-key generation. Also consider roles that enforce a dual-control security policy; for example, permit one individual/role to register a hash and another individual/role to register a public key, have different individuals/roles for obtaining and installing the individual shares of the master key, and so forth.

See the guidance portion of Chapter 2 in the *IBM 4758 CCA Basic Services Reference and Guide* and the description of the Master_Key_Process and the Master_Key_Distribute verbs in the same chapter.

2. Install a unique 1- to 16-byte Environment ID (EID) of your choice into each node.

From the **Crypto Node** pull-down menu, select **Set Environment ID**, enter the identifier, and select **Load**. Use only these characters in an environment identifier (EID): A...Z, a...z, 0...9, and “@” (X'40'), space character (X'20'), “&” (X'26'), and “=” (X'3D').

You should enter a full 16-character identifier. For ‘short’ identifiers, complete the entry with space characters.

3. Initialize the master-key-sharing “m” and “n” values in the source and target nodes. These values *must* be the same in the source and the target node. “n” is the maximum number of shares while “m” is the minimum number of shares that must be installed to reconstitute the master key in the target node.

From the **Crypto Node** pull-down menu, select **Share Administration**, and then select **Set number of shares**, enter the values, and select **Load**.

4. At the different nodes, generate these keys and have each public key certified by the Share-Administration (SA) key. You can use the utility's sa.db data base to transport the keys and the certificates.

Share Administration (SA)

This key is used to certify itself and the following keys. You must register the hash of the SA public-key, and the public key itself, in the SA, the source, and the target nodes.

When the SA key is created, the utility will supply an 8-byte/16-hex-character value that is a portion of the hash of the SA key. *Be sure to retain a copy of this value.* You will need this value to confirm the hash value recorded in the data base to register the SA public-key at the source and target nodes.

Coprocessor Share Signing (CSS)

This key is used to sign shares distributed from the source node. The private key is retained within the source node.

Coprocessor Share Receiving (CSR)

This key is used to receive a share-encrypting key into the target node. The SA-certified public CSR key is used at the source node to wrap (encrypt) the share-encrypting key that is unique for each share. The private key is retained within the target node.

Generate the Key Pairs: SA, CSS, and CSR

From the **Crypto Node** pull-down menu, select **Share Administration**, select **Create Keys**, and one of **Share Administration, C... S... S... Key**, or **C... S... R... Key**, then select **Create**.

You also will need to supply key labels for the CSS and CSR keys that are retained in the source and target nodes. For example, 'IBM4758.CLONING.CSS.KEY' and 'IBM4758.CLONING.CSR.KEY'; the labels that you use must not conflict with other key labels used in your applications.

When generating the CSR key at the share-receiving node, also obtain the serial number of the Coprocessor. From the **Crypto Node** pull-down menu, select **Status**. You must enter the serial-number value when certifying the CSR key.

5. Register the SA public-key in the Coprocessor at the SA, source, and target nodes. This is a two-step process that should be done under a dual-control security policy.

One individual should install the SA public-key hash. From the **Crypto Node** pull-down menu, select **Share Administration**, select **Register share administration**, and select **SA key hash**. You will enter the hash value obtained during SA key creation.

The other individual should install the actual SA public-key. From the **Crypto Node** pull-down menu, select **Share Administration**, select **Register share administration**, and select **SA key**. By default, the public-key information is in the sa.db file.

6. Take the CSS key and the CSR key to the SA node and have the keys certified.

From the **Crypto Node** pull-down menu, select **Share Administration**, select **Certify Keys**, and one of **C... S... S...**, or **C... S... R...**

For the CSR key, you will need to supply the serial number of the target Coprocessor as a procedural check that an appropriate key is being certified. Your procedures should include communicating this information in a reliable manner.

7. At the source node, have authorized individuals sign on to the role that permits them to obtain their shares. At least "m" shares must be obtained. These will be shares of the current master key.

From the **Crypto Node** pull-down menu, select **Share Administration**, select **Get share**, and select the share number to be obtained. Observe the serial numbers and data base identifiers. When these are agreed to be correct, select **Get Share**. The share information will be placed by default into the csr.db file and will obtain the CSR key-certificate, by default, from the sa.db file.

Obtain current-master-key validation information for use later at the target node. From the **Master Key** pull-down menu, select **Verify**, select **Current**.

8. At the target node, have authorized individuals sign on to the role that permits each of them to install his or her share. At least “m” shares must be installed to reconstitute the master key into the new-master-key register.

From the **Crypto Node** pull-down menu, select **Share Administration**, select **Load share**, and select the share number to be installed. Observe the serial numbers and data base identifiers and when these are agreed to be correct, select **Install share**. The share information will be obtained by default from the csr.db file and the CSS key certificate will be obtained by default from the sa.db file.

When “m” shares have been loaded, verify that the key in the new-master-key register is the same as the current master-key in the source node (when the shares were obtained). On the target node, from the **Master Key** pull-down menu, select **Verify**, select **New**.

9. When it is confirmed through master-key verification that the master key has been cloned, an authorized individual can set the master key. This action deletes any old master-key and moves the current master-key to the old-master-key register. Application programs that use keys encrypted by the master key can be impacted by this change, so be certain that setting of the master key is coordinated with the needs of your application programs. From the **Master Key** pull-down menu, select **Set**.

Managing Key Storage

The Cryptographic Node Management utility allows basic key-storage management for keys. These utility functions do not form a comprehensive key-management system. Application programs are better suited to perform repetitive key-management tasks.

Key storage is a repository of keys that you access by key label using labels that you or your applications define. DES keys and “PKA” (RSA) keys are held in separate storage systems. Also, the Coprocessor has a very limited internal storage for RSA keys. The Coprocessor-stored keys are not considered part of key storage in this discussion.

This section describes how to:

- Create or initialize key storage
- Reencipher stored keys
- Delete a stored key
- Create a key label.

Note: The utility displays a maximum of 1,000 key labels. If you have more than 1,000 key labels in key storage, use an application program to manage them.

How to Create or Initialize Key Storage

To create or initialize key storage for your DES or PKA keys:

1. From the **Key Storage** pull-down menu, select **DES Key Storage** or **PKA Key Storage**; a sub-menu is displayed.
2. From the sub-menu, select **Initialize**; the Initialize DES Key Storage or the Initialize PKA Key Storage panel is displayed.

3. Enter a description for the key-storage file, if desired.
4. Select **Initialize**; you are prompted to enter a name for the key-storage file.

Note to AIX Users: The location you set for key storage *must* match the location defined in the AIX object data manager (ODM). See “How to Configure the Support Program” on page 3-3 and the use of the ODMGET and CSUFKEYS utilities.

Note to Windows NT/Windows 2000 Users: The location you enter for key storage must match the information that you provided during loading of the CCA support program software. These locations are recorded in the Windows NT/Windows 2000 Registry. Look in the Registry for DES.KEY and PKA.KEY.

5. Enter a name for the file and save it. The key-storage file is created on the host.

Note: If a file with the same name exists, you are prompted to verify your choice because initializing the key storage modifies the file, and if it had any keys, these would be erased.

How to Reencipher Stored Keys

To reencipher the keys in storage under a new master-key:

1. From the **Key Storage** pull-down menu, select **DES Key Storage** or **PKA Key Storage**; a sub-menu is displayed.
2. From the sub-menu, select **Manage**; the DES Key Storage Management or the PKA Key Storage Management panel is displayed. The panel lists the labels of the keys in storage.
3. Select **Reencipher...**; the keys are reenciphered using the key in the current master-key register.

How to Delete a Stored Key

To delete a stored key:

1. From the **Key Storage** pull-down menu, select **DES Key Storage** or **PKA Key Storage**; a sub-menu is displayed.
2. From the sub-menu, select **Manage**; the DES Key Storage Management or the PKA Key Storage Management panel is displayed. The panel lists the labels of the keys in storage.

You can set the filter criteria to list a subset of keys within storage. For example, entering “*.mac” as the filter criterion and refreshing the list limits it to keys with labels that end in “.mac.” (The asterisk is a wildcard character.)

3. Highlight the key label for the key to be deleted.
4. Select **Delete...**; a confirmation dialog box is displayed.
5. Select **Yes** to confirm; the stored key is deleted.

How to Create a Key Label

To create a key label:

1. From the **Key Storage** pull-down menu, select **DES Key Storage** or **PKA Key Storage**; a sub-menu is displayed.
2. From the sub-menu, select **Manage**; the DES Key Storage Management or the PKA Key Storage Management panel is displayed. The panel lists the labels of the keys in storage.

You can set the filter criteria to list a subset of keys within storage. For example, entering “*.mac” as the filter criterion and refreshing the list limits it to keys with labels that end in “.mac.” (The asterisk is a wildcard character.)
3. Select **New**; you are prompted to enter a key label.
4. Select **Load**; the key label is loaded into storage.

How to Create and Store Primary DES KEKs

Key-encrypting keys (KEKs) are encrypted under the master key and stored in key storage for local use. Key parts used to create a KEK can be randomly generated or entered as clear information. The parts can also be saved to disk or diskette in the clear for transport to other nodes or for re-creating the local KEK.

Note: The Cryptographic Node Management utility only supports DES KEKs for the transport of keys between nodes. Applications can use the CCA API to furnish the services needed for public-key-based key distribution.

To work with a KEK (or other double-length operational key):

1. From the **Keys** pull-down menu, select **Primary DES Key-Encrypting Keys**; the Primary DES Key-Encrypting Keys panel is displayed.
2. At any time you can select **New** to clear all data fields and reset all the radio buttons to their default settings.
3. Select the radio button for the desired key-part to be entered: **First Part**, **Middle Part**, or **Last Part**.
4. Enter data in the **Key Part** fields by using one of the following processes:
 - Select **Open...** to retrieve pre-existing Key Part, Control Vector, and Key Label data previously stored on disk using the **Save...** command.
 - Select **Generate** to fill the Key Part fields with Coprocessor-generated random numbers.
 - Manually enter data into the “Key Part” fields; each of the Key Part fields accepts four hexadecimal digits.
5. Select a control vector for the key:
 - To use a default KEK control-vector, select the appropriate **Default Importer** or **Default Exporter** radio button.
 - To use a custom control-vector, select the **Custom** radio button. In the Control Vector fields, enter the left or right half of a control vector for any double-length key. Note that the key-part bit (bit 44) must be on and that each byte of the control vector must have even parity. For detailed information about control vectors, refer to Appendix C of the *IBM 4758 CCA Basic Services Reference and Guide*.

- 1 6. Enter a key label to identify the key token in key storage.
- 1 7. Select **Load** to load the key part into the Coprocessor and store the resulting key token into key storage; select **Save...** to save the unencrypted Key Part and its associated Control Vector and Key Label values to disk.
- 1 8. **Save** to disk or **Load** to key storage the remaining key-part information by following Step 3 on page 5-23 to Step 7. Be sure to use the same key label for each part of a single key.

Using the CNI Utility to Establish Other Nodes

By creating a CNI list for the Cryptographic Node Initialization (CNI) utility, you can load keys and access-control data stored on disk into other cryptographic nodes without running the Cryptographic Node Management utility on those target nodes.

To set up a node using the CNI utility:

1. Start the Cryptographic Node Management utility on an established node.
2. Save to the host or portable media (like a floppy disk) the access control and keys you want to install on other nodes. When you run the CNI utility on the target node (Step 10), it searches the identical directory path for each file. For example:
 - If you save a user profile to the established node directory `c:\IBM4758\profiles`, the CNI utility will search the target node directory `c:\IBM4758\profiles`.
 - If you save a user profile to the floppy disk directory `a:\profiles`, the CNI utility will search the target node directory `a:\profiles`.
3. From the **File** pull-down menu, select **CNI Editor**; the CCA Node Initialization Editor panel is displayed. See Figure 5-4.

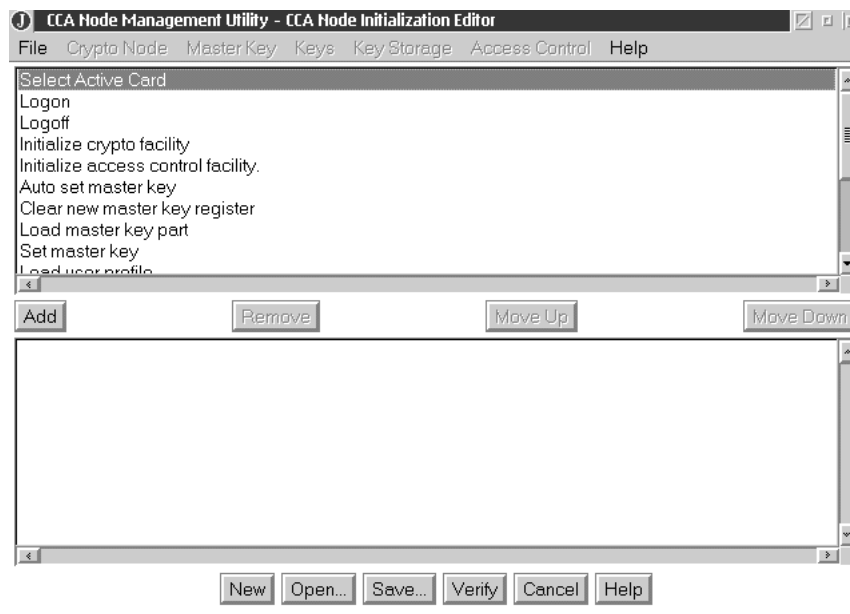


Figure 5-4. The CCA Node Initialization Editor Panel

The list in the top portion of the panel displays the functions able to be added to the CNI list; the bottom portion lists the functions included in the current CNI list. The CNI list can perform the following functions:

- Initialize the cryptographic facility (Coprocessor)
- Synchronize the clock-calendar
- Load or delete roles and user profiles
- Logon and logoff to the cryptographic node
- Load master-key parts
- Generate a random master-key
- Set the master-key registers
- Auto-set the master-key registers
- Clear the new-master-key register
- Load primary KEK parts
- Initialize storage for DES keys and PKA keys.

4. Add the functions you want. To add a function to the CNI list:

- a. Highlight it.
- b. Select **Add**; the function is added to the CNI list.

Note: If the function you choose loads a data object—like a key part, key-storage file, user profile, or role—you are prompted to enter the file name or the ID of the object to be loaded.

5. Using the **Move Up** and **Move Down** buttons, organize the functions to reflect the same order you follow when using the Cryptographic Node Management utility. For example, if you are loading access-control data, you must first log on as a user with the authority to load access-control data.

6. Select **Verify** to confirm that objects have been created correctly.

7. Select **Save...**; you are prompted to choose a name and directory location for the CNI-list file.

8. Save the CNI-list file; the list file *does not* contain the data objects specified in the CNI list.

9. Copy the files needed by the Cryptographic Node Initialization utility onto target host directory locations that mirror their location on the source host. If you saved the files to portable media, insert the media into the target node.

10. From the target node, run the list using the Cryptographic Node Initialization utility:

- On AIX systems, enter **csufcni** *listfile_name* on the command line.
- On Windows NT/Windows 2000 systems:
 - Change directory to **\program files\ibm\4758\cnm**
 - Enter **csuncni** *listfile_name* on the command line.

If the CNI list includes a logon, enter **csufcni**, **csuecni**, or **csuncni** on the command line (without specifying a filename). The utility Help text describes the syntax for entering an ID and passphrase.

The Cryptographic Node Initialization utility loads files to the Coprocessor from the host or portable media, as specified by the CNI list.

Chapter 6. Observations on Secure Operations

This chapter offers a series of observations about the setup and use of the IBM 4758 CCA cryptographic node that you may consider in order to enhance secure operations. Topics include:

- Ensuring code levels match and IBM CCA code is installed
- Access controls
- Cryptographic keys
- PIN data
- Status data
- RS-232 port
- Master-key cloning
- Sample access-control regimes.

Ensuring Code Levels Match and IBM CCA Code is Installed

The level of the CCA code in the host system should match that used within the Coprocessor. You can download and install code from IBM's Web site. See Chapter 2, "Obtaining Coprocessor Hardware and Software" and Chapter 3, "Installing the Support Program" for details.

Following the instructions in Chapter 4, "Loading Software into the Coprocessor" and the README information for your copy of Coprocessor code, install the code into the IBM 4758. Use the VA command of the CLU utility to obtain and validate a signed Coprocessor response. Be sure that the segment 2 and segment 3 owner identifiers are valued to 2. A segment 3 owner identifier other than 2 indicates that the code is not the standard-product IBM CCA code. (If your code incorporates a User-Defined Extension (UDX, custom code), an extended form of CCA could be present.) If segment 2 has an owner identifier of 6, there is the possibility of loading a code-debugging probe that can compromise the security of any code running in segment 3.

You should ensure that the host-side code supplied with the IBM CCA Support Program has not been altered from that obtained from IBM's Web site.

Access Controls

The access-control system and the grouping of permissible commands¹ that you can employ are designed to support a variety of security policies. In particular, you can set up the CCA node to enforce a dual-control, split-knowledge policy. Under this policy, once the node is fully activated, no one person should be able to cause detrimental actions other than a denial-of-service attack. To implement this policy, and many other approaches, you will necessarily have to limit your use of certain commands. Therefore, as you design your application, you should consider the commands you must enable or restrict in the access-control system and the implications to your security policy. See also Appendix A, "CCA Access-Control

¹ Commands, verbs, and the access-control system are described in the first chapters of the *IBM 4758 CCA Basic Services Reference and Guide*.

Commands” for a table of commands with general guidance in the right-hand columns.

The following sections describe:

- Locking the access-control system
- Changing a passphrase
- Defining roles and profiles.

Locking the Access-Control System

For secure operation after performing any initializing processes, consider “locking” the access-control system. You can render the access-control system unchangeable by deleting any profile that would allow use of the Access Control Initialization command (X'0112', invoked with the Access_Control_Initialization verb and INIT-AC keyword), and the Delete Role command (X'0118', invoked with the Access_Control_Maintenance verb and DEL-ROLE keyword). Without these commands, further changes to the access-control roles are not possible.

Before the CCA node is put into normal operation, the access-control setup can be audited through the use of the Access_Control_Maintenance and Cryptographic_Facility_Query verbs. If for any reason the status response is not as anticipated, the node should not be activated for application purposes.

Note: With authority to use either the Initialize Access Control or Delete Role commands, one can delete the DEFAULT role. Deleting the DEFAULT role will cause the automatic recreation of the initial DEFAULT role. The initial DEFAULT role permits setting up any capabilities. Users with access to these commands have unlimited authority through manipulation of the access-control system.

Passphrase Considerations

The passphrase used to authenticate access to a profile is not communicated out of the DLL or shared library you call with the Logon_Control verb. Rather, the passphrase is hashed to form a cryptographic key that is used to pass the profile identifier and other information to permit the Coprocessor to validate access to the profile.

When you change a passphrase with the Access_Control_Initialization verb, use the **PROTECTD** keyword. This causes the passphrase to be encrypted within the DLL or shared-library layer before it is communicated to the Coprocessor. This can block a lower-level sniffer program or the CCA trace facility² from capturing the new, clear passphrase.

If a role contains permission to change a passphrase, the passphrase of *any* profile can be changed. You should consider if passphrase changing should be permitted and, if so, which role(s) should have this authority (command X'0114').

If any user reports an inability to log on, this should be reported to someone other than (or certainly in addition to) an individual with passphrase-changing permission.

² The IBM CCA Support Program prior to Release 2.41 incorporates a trace facility that can be used by IBM product development in aiding the diagnosis of obscure problems. This trace facility can capture the clear passphrase information as it flows in the host system. This and other techniques could be used by an adversary to capture clear passphrase information.

Roles and Profiles

The access-control system, which is discussed in the opening pages of Chapter 2 of *IBM 4758 CCA Basic Services Reference and Guide*, permits users to define roles and profiles as suits their operation and security needs. Roles and profiles you might consider include:

Table 6-1 (Page 1 of 2). Example Roles

Setup	<p>A Setup role can be defined that enables loading of required roles, profiles, and other special values such as the Environment ID (EID), Function-Control Vector (FCV), set up of the master-key shares-cloning m-of-n values, and registration of a public key(s) for later use in key distribution.</p>
Administrator	<p>You can establish an Administrator role(s) with extensive supervisory capabilities. The administrative roles could be permitted to:</p> <ul style="list-style-type: none"> • Change the passphrase of any profile and reset the failure count of any profile (Access_Control_Initialization verb). <p>An individual entrusted with these responsibilities can log on to any role by changing the passphrase of an associated profile and thereby gain the permissions of any role. However, he might not be able to restore the passphrase of the normal user of the profile since in a secure installation he should not know another user's passphrase. You can address this problem in these ways:</p> <ul style="list-style-type: none"> • Disabling a role that permits passphrase changing, or • At a minimum, ensuring that any suspected authentication problems are reported to someone other than the administrator(s) having use of roles permitting passphrase changing. <p>Note: You are advised to set up a duplicate administrator role and associated profiles with a different expiration date to insure that you will have access to those services appropriate to the administrator. This may give you an opportunity to recover should the primary administrator make an error that cannot be rectified.</p>
SO1, Security Officer 1	<p>Security Officer 1's role(s) could be permitted to:</p> <ul style="list-style-type: none"> • Randomly generate a master key • Import a key-encrypting key.
SO2, Security Officer 2	<p>Security Officer 2's role(s) could be permitted to:</p> <ul style="list-style-type: none"> • Set a master key • Import keys. <p>Note: If you employ introduction of keys in parts (Key_Part_Import and/or Master_Key_Process verbs; see "Cryptographic Keys" on page 6-4), the first-part and second-part permissions should be assigned to SO1 and SO2, respectively. See also "Clear Key-Parts" on page 6-6.</p>

Table 6-1 (Page 2 of 2). Example Roles

Default	You must have a Default role. When a host thread is not logged on, requests from such a thread are performed based on the permissions set in the default role. You should enable only those commands necessary for normal operations. At a maximum, only those functions specifically required should be enabled. All sensitive or unusual requirements should be processed following a logon to an appropriate profile (and thus its role).
Application User _n	<p>As required, “n” application-specific roles and associated profiles should be established for processing portions of applications with security requirements different from those permitted under the Default role.</p> <p>For example, enabling any of the key export verbs could lead to the possibility that keys are released to an adversary. Such operations are candidates for selective enablement under control of a specific role.</p>

In all cases, only those commands actually needed to accommodate the permitted applications should be enabled.

Cryptographic Keys

Cryptographic keys are generally passed across the CCA interface as encrypted objects in key-token data structures. Rogue processes on your host system might be able to capture a copy of such keys, or the contents of the key-storage data set may be copied. You must rely on your operating-system security, system-operational security, and physical security to counter any threat from an encrypted-key copy. Be careful not to let a rogue process be able to make use of the encrypted key. Your environmental security policy should consider how rogue processes could make use of (a copy of) an encrypted key. You certainly must consider the handling of any clear keys.

b
b
b

Keys are further discussed under these topics:

- CCA “asymmetric” DES keys
- Clear key-parts
- Key export
- Unwrapping keys
- Operations with clear keys
- Using DES “replicated” keys
- RSA keys.

b
b

CCA Asymmetric DES keys

With CCA, you can often make use of a unique capability afforded through the CCA control vector and command architecture. CCA enables DES keys to have asymmetric properties. Using MAC/MACVER, ENCIPHER/DECIPHER, IMPORTER/EXPORTER, PINGEN/PINVER, and IPINENC/OPINENC key types, you can separate which systems and processes can “reverse” various cryptographic functions.

MAC/MACVER

A node that has a MAC-class key can both generate and verify a DES MAC value. A CCA node having only the key with MACVER properties is unable to create an authentication code (a MAC) with the key. Thus, data recipients who receive only a MACVER key can be enabled to validate data but are prevented from producing a MAC on data potentially altered to their advantage.

Note also that a DES MAC is computed by enciphering the cleartext data. You also need to ensure that an adversary is denied access to enciphering processes with the key used in the MAC computations. For this reason, consider use of the MAC and MACVER keys rather than the DATA-class keys. By default, DATA-class keys perform in both encipher and MAC generation and verification operations.

ENCIPHER/DECIPHER

b
b
b

You can separate the ability to reverse a DES ciphering process. Use an ENCIPHER key at the data-encryption node and only supply the data-decryption node with the DECIPHER form of the ciphering key.

You may also find uses for enciphering data where you want to disallow the possibility that the data is ever deciphered. You can determine the equivalence of two copies of source data by comparing their enciphered value. Thus you can store an enciphered copy of data and determine later that other data is not equivalent without revealing the clear value of the original data. Of course, a hash process can give the same effect, but the IBM 4758 can perform DES encryption at up to 22.5 megabytes per second.

1
1
1

Consider the use of the CIPHER-class keys rather than the more general DATA-class keys since the former have reduced capabilities and thus offer fewer opportunities for misuse.

IMPORTER/EXPORTER

You use a key in these key classes to set up a one-way key-distribution channel. In fact, it is generally considered inappropriate to have the same key-value encrypted as both an IMPORTER and as an EXPORTER on the same CCA node. You can use the functionality of the Key_Generate verb and the one-way key-distribution channel to distribute CCA "asymmetric" DES keys to node pairs.

For example, a data originator can encipher data and be sure that no one can decipher the data on his node through the use of an ENCIPHER-class key. The DECIPHER-class copy of the key, probably with the CCA export-allowed control-vector bit turned off, can be sent over the one-way key-distribution channel to another node. Only there can the data be deciphered.

As another example, a key-distribution center can originate and distribute a no-export-allowed MAC key to one node and the matching MACVER key, also with the no-export-allowed attribute, can be sent to another node. In this scenario (and if the CCA master keys are managed and audited in a secure manner), the MAC verification node has no means of producing a valid MAC on altered data.

PINGEN/PINVER

You can segregate the ability to create a PIN value from the ability to validate a PIN value (and PIN offsets, PVV values, and so forth).

OPINENC/IPINENC

As with one-way key-distribution channels, you can set up one-way encrypted PIN-block distribution channels. This can enable you to further segregate which nodes in your network can perform various forms of PIN processing.

Clear Key-Parts

A traditional means for instantiating a cryptographic key is to have two or more users each install a “key part.” The key parts are exclusive-ORed together to form the final key. CCA supports this option with the Key_Part_Import and Master_Key_Process verbs. You can force the separation of key-part installation into two groups by enabling the first-part capability and the key-part-combine capability in different roles. (And you can use different roles for processing master keys versus other key types.)

1 Release 2.41 added options to the Key_Part_Import verb to provide additional
1 separation in functional capability. (These new options are not employed in the
1 CNM utility.) The preexisting Combine Key Parts command (X'001C', invoked
1 with the MIDDLE and LAST keywords) will process key-part information as well as
1 complete a key by turning off the key-part control-vector bit. Two new options are
1 added:

- 1 • Add Key Part (X'0278') is used to exclusive-OR additional information into an
1 incomplete key
- 1 • Complete Key Part (X'0279') is used to turn off the key-part control-vector bit.

1 The new commands enable you to designate an individual with the authority to
1 accept a key without providing that individual with the possibility of modifying the
1 value of the key.

1 **Pre-Exclusive-OR:** You should recognize that additional information such as
1 variant or control-vector value(s) can be exclusive-ORed into a key-encrypting key
1 such that other keys processed with the modified key-encrypting key can be
1 assigned alternative control vectors and therefore functionality. This
1 “pre-exclusive-OR” technique can be valuable when exchanging keys with non-CCA
1 systems.³ However an adversary, if permitted to employ the technique, could
1 change the control vector of a key to his advantage. To counter this threat, you
1 can separate key-part loading responsibilities so that an individual can oversee the
1 operations and “complete” a key without having the authority to add information to
1 the key.

1 **Clear Key-Part Interception:** Note that the key-part information flows in the clear
1 through your host system.⁴ In some cases you may view this as an unacceptable
1 risk. In these cases consider alternatives such as:

Random generation of master keys.

If you need to backup the master key or have the same master key in an additional Coprocessor(s), use master-key cloning to securely transfer the value of the master key to additional Coprocessors.

Random key-generation and RSA-based key-distribution.

Distribute RSA-encrypted, randomly generated DES data or DES key-encrypting keys to the node where the key should be instantiated. With CCA and this strategy, you will not need key parts and you will not need secrecy. (You

1 ³ See the section “Changing Control Vectors with the Pre-Exclusive-OR Technique” in Appendix C of the *IBM 4758 CCA Basic*
1 *Services Reference and Guide*.

1 ⁴ Prior to Release 2.41, the IBM CCA Support Program incorporates a trace facility that can be used by IBM product development in
aiding the diagnosis of obscure problems. This trace facility captures the clear key-part information as it flows in the host system
Windows, OS/2, or AIX environments. This and other techniques could be used by an adversary to capture clear key-part
information.

should, however, continue to use two-channel distribution techniques to ensure integrity of the public-key distribution. This is true even when certificates are in use; you need to provide integrity for the top-level public key.)

Key Export

You should have a concern for the export of keys from your system. Take special care in the enablement of the PKA_Symmetric_Key_Generate verb and the three key-export verbs, Data_Key_Export, Key_Export, and PKA_Symmetric_Key_Export. Note especially that the verbs PKA_Symmetric_Key_Export and PKA_Symmetric_Key_Generate permit the export of selected classes of keys under “any” public key. You need to ensure that the target nodes are legitimate and that only appropriate processes have use of these verbs, EXPORTER keys, and public keys.

Consider taking maximum advantage of the export-allowed control-vector bit. By switching this bit off, you can prevent a key from being exported.

Note: Master-key-encrypted RSA private keys or retained RSA private keys cannot be exported from a CCA node.

Unwrapping Keys

An RSA private key that is used in a symmetric key exchange to unwrap (decrypt) a DES key should not normally be allowed to perform digital signature signing. This is because the Digital_Signature_Generate verb could be used with the ZERO-PAD keyword to reveal the DES key. To prevent this from happening, generate RSA keys used for symmetric key-exchange with their key-usage flag bits set to KM-ONLY. The key-usage flag bits should not be set to KEY-MGMT as this allows the key to be used for both symmetric key exchange and digital signature signing.

Operations with Clear Keys

Remember that the following CCA verbs operate with keys in the clear. Their use should be carefully considered.

CSNDSYX
PKA_Symmetric_Key_Export

A clear, unprotected public key is entered under which DATA keys can be enciphered. This operation can be disallowed through the access-control system.

This is a potentially insecure operation in that any DATA key having the EXPORT-ALLOWED bit on can be exported to the owner of the associated private key.

CSNDSYG
PKA_Symmetric_Key_Generate

A clear, unprotected public key is entered under which a freshly generated KEK or DATA key can be created. This operation can be disallowed through the access-control system.

This is a potentially insecure operation if you set up a key-distribution channel with an inappropriate public key. Be sure that you know who has access to the associated private key.

	CSNBCKI	Either 8 or 16 bytes of clear information can be accepted to be returned as an encrypted DATA key. This operation can be disallowed through the access-control system. The clear key information could be intercepted as it is transmitted to the Coprocessor. Consider freshly generating a key using Key_Generate.
	Clear_Key_Import	
	CSNBCKM	
	Multiple_Clear_Key_Import	
1	CSNBKPI	This verb requires use of two commands (using the FIRST , with MIDDLE and LAST keywords), or three commands (using the FIRST , with ADD-PART and COMPLETE keywords) to complete the establishment of a productive key of any type. The key information is passed in the clear. These operations can be disallowed through the access-control system. The access controls can enforce a dual-control policy, but the key components (parts) still pass across the general interface in the clear. As an alternative, consider use of PKA_Symmetric_Key_Import and Key_Import to receive keys from another source. Note that an adversary might be able to change the value of a key by employing use of the MIDDLE or ADD-PART keywords. If the key were for an IMPORTER or EXPORTER, this could be used later to alter the control vector of an imported or exported key. This technique is sometimes viewed as a legitimate means for altering control vectors and is referred to as the <i>pre-XOR</i> technique. (See also "Pre-Exclusive-OR" on page 6-6 and "Clear Key-Part Interception" on page 6-6.)
1	Key_Part_Import	
1		
1		
1		
1		
1		
1		
1		
1		
	CSNBMKP	Use of the FIRST , MIDDLE , and LAST keywords employs clear data to establish the value of a master key. These operations can be disallowed through the access-control system. The preferred means to establish a master key is through random generation (RANDOM keyword) or through the <i>master-key cloning</i> process. (See also "Clear Key-Part Interception" on page 6-6.)
	Master_Key_Process	
1		
1		
	CSNDDSG	The ZERO-PAD option allows a "hash" to be specified which can be as long as the modulus of the key. If the "hash" is in fact a public-key-encrypted key, the key can be recovered in the clear. It is recommended that RSA keys intended for key management be restricted to that usage by the specification of KM-ONLY in PKA_Key_Token_Build and PKA_Key_Generate.
	Digital_Signature_Generate	

b Using DES Replicated Keys

A "replicated key" is defined as a double-length DES key having equal left and right halves. Such a key performs as a single-length key. Since CCA always uses double-length key-encrypting keys and PIN-processing keys, it is sometimes advantageous to generate or install replicated keys in order to inter-operate with non-CCA systems which use single-length DES keys.

Be careful in permitting the generation and use of replicated keys as overcoming the work-factor to attack single-length DES keys may be within the capability of certain adversaries. You can block the generation of replicated DES keys in the

Key_Generate and the Diversified_Key_Generate verbs by not enabling optional commands.

1 Beginning with Release 2.41, the Key_Import, Key_Export, Data_Key_Import, and
1 Data_Key_Export verbs no longer permit use of a replicated key-encrypting key to
1 import or export a non-replicated key unless special permission is granted through
1 the use of the “unrestrict” commands: X'0276', X'0277', X'027B', and X'027C'.

b **Generating RSA Keys**

b When generating RSA key-pairs, you must consider these issues:

- b 1. Regeneration data
- b 2. Attributes for signature and key management
- b 3. Retained keys.

b **Regeneration Data**

b When an application calls the PKA_Key_Generate verb it can specify
b *regeneration_data* which will seed the random number generator. If the same
b regeneration data is supplied in the future, the same RSA key-pair will be
b generated. This can be useful in a development environment but is inappropriate in
b a production environment.

b In the current implementation, the access-control system does not provide a way to
b block use of regeneration data. Therefore, you must inspect application programs
b which employ the PKA_Key_Generate verb to ensure appropriate operation. For a
b production environment, the *regeneration_data_length* variable should be zero.

b **Attributes for Signature and Key Management**

b The skeleton key-token that an application provides when calling the
b PKA_Key_Generate verb assigns usage attributes to the generated private key.
b You can assign attributes that only permit the key to perform in digital signature
b generation or only in unwrapping (“importing”) symmetric keys. Or you can assign
b attributes that enable the key to generate both digital signatures and unwrap keys.

b If you use the Digital_Signature_Generate verb with the **ZERO-PAD** option and
b supply a wrapped symmetric key, the verb will decrypt the wrapped key and an
b application can then recover the symmetric key in the clear. This most likely is
b undesirable. Generally, private keys intended for use in key-management
b applications should be generated with the **KM-ONLY** attribute to block the use of
b the key in the Digital_Signature_Generate verb. For private keys used to generate
b digital signatures, consider applying the **SIG-ONLY** attribute.

b **Retained Keys**

b In some applications it is required that an RSA private key not be released in any
b form from the device in which it is generated. Use of the **RETAIN** keyword with the
b PKA_Key_Generate verb causes the Coprocessor to not output the private key.
b (Alternative choices permit the key to be returned to the application wrapped under
b the master key or a transport key, or with access-control permission, returned in the
b clear.)

b You can confirm that a key has been generated and retained in the Coprocessor
b through the use of the Retained_Key_List verb and the verification of a digital
b signature made with the retained key. Since there is no means to import an RSA
b private key into retained-key storage, the list of key labels returned from the

b Retained_Key_List verb demonstrates the existence of a particular named key.
b You can confirm that the same label does not exist in host-system key-storage
b through the use of the PKA_Key_Record_List verb.

PIN Data

A Personal Identification Number (PIN) is generally passed across the CCA interface as an encrypted object in an encrypted-PIN-block. Generally all verbs protect PIN values through encryption. The exceptions are:

CSNBCPE Clear_PIN_Encrypt	Encrypts a clear-PIN value and returns the result under an OPINENC class key. This operation can be disallowed through the access-control system. Unrestricted usage can permit the construction of a dictionary of encrypted PIN values.
CSNBPGN Clear_PIN_Generate	Generates the PIN for a given account number. This operation can be disallowed through the access-control system. Unrestricted usage permits the generation of PIN numbers for the specified account number(s), using information that can be known to an adversary.

Status Data

Status is returned from the CCA application through the use of the Access_Control_Maintenance and Cryptographic_Facility_Query verbs. An adversary with access to the computing system could alter Coprocessor status responses.

Note also that certain status information can be obtained from the Miniboot component of the Coprocessor through the use of the Coprocessor Load Utility (CLU). This response is signed and can be validated using the CLU utility.

RS-232 Port

All CCA input and output is via CP/Q++. With release 2.40, the embedded control program, CP/Q++, provides a device driver for the RS-232 communications port which is integral to the Coprocessor. However, the standard CCA application program makes no use of the port and therefore the port is functionally inert. No information from or to the standard implementation of CCA will pass over the RS-232 port interface.

Master-Key Cloning

If you employ master-key cloning, then the distribution of shares needs to be accommodated, perhaps with a unique role and profile for the individual permitted to process share_n. Registering the public key of the authorization node should be split between two users such as SO1 and SO2. See Table 6-1 on page 6-3.

Sample Access-Control Regimes

The CCA access-control system is quite flexible so as to accommodate a wide variety of needs. Your task is to balance simplicity of operation against the requirements for a secure installation. This section discusses several cases as an introduction to establishing your access-control regime.

b
b

- Considerations for a digital-signing server
- Considerations for a secured code-signing node.

b **Digital-Signing Server**

b The European Union has issued a directive outlining the requirements for
b equipment and software to qualify for use in certain Public Key Infrastructure (PKI)
b applications. In compliance with the directive, individual countries such as
b Germany have issued detailed requirements for evaluations under the Common
b Criteria. In IBM's opinion, the IBM 4758 Model 002 operating with the CCA
b Support Program can be employed as a "hardware security module" (HSM) in PKI
b CA, RA, OCSP responder,⁵ and time-stamp authority applications in compliance
b with the directive.

b The remainder of this section outlines considerations for use of the IBM 4758
b consistent with the high-security requirements of these application areas. The IBM
b 4758 with CCA offers many capabilities. If you choose to deviate from some of the
b guidelines in this section, for example to achieve greater flexibility in operation, be
b sure to review the earlier portions of this chapter for guidance.

b The guidelines are organized under these topics:

- b • Application environment
- b • Threat considerations
- b • Security objectives
- b • Application requirements
- b • Policy requirements
- b • Operational requirements and plan.

b **Application Environment**

b The application environment includes a computing system and the surrounding
b policies, personnel, and physical security to address Public Key Infrastructure CA,
b RA, OCSP responder, or time-stamping services. An application program must be
b selected which employs CCA and the Coprocessor and which implements the
b desired services. The application program must be reviewed against the guidelines
b discussed in the section "Application Requirements" on page 6-14. The
b organization must install both an IBM 4758 Coprocessor (Model 002⁶ or the
b equivalent feature #4963 on an IBM eServer pSeries system) and the CCA Support
b Program as described earlier in this manual. The organization will need to observe
b the requirements discussed in sections "Policy Requirements" on page 6-14 and
b "Operational Requirements and Plan" on page 6-16.

b The security functions of the Coprocessor do not rely on the correctness of the host
b software or the fact that this software is not tampered with or bypassed. But of
b course for the Coprocessor to be useful, functions of the host system need to
b execute correctly. For example, if a human user wants a document to be signed, it
b is a requirement that the host software ensure that:

- b 1. The document displayed to the user is represented by a hash of the document,
b and
- b 2. The hash is forwarded unaltered to the Coprocessor for inclusion in the digital
b signature.

b ⁵ CA: Certification Authority; RA: Registration Authority; OCSP responder: Open Certificate Status Protocol responder.

b ⁶ An IBM 4758 Model 023 may also be used if the lower level of tamper detection is satisfactory in your application environment.

Application Requirements

The selected application must:

1. Employ the CCA interface as described in the IBM 4758 publications:
 - Release 2.41 *IBM 4758 CCA Basic Services Reference and Guide*
 - Release 2.41 *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual*, this manual.
2. Generate an RSA key-pair which meets these requirements:
 - a. Has an appropriate key-length, $1024+256s$, $0 \leq s \leq 4$
 - b. Flags the key for use only in digital signature operations⁷
 - c. Extends the skeleton key-token to obtain a self-signature that incorporates the Coprocessor serial number and application-supplied information.⁸
 - d. Is retained within the Coprocessor⁹
 - e. Is not generated with reference to any "regeneration data"¹⁰
 - f. Makes known the key label used to access the private key so that an auditor may confirm that a retained key is in use.¹¹
3. Generate digital signatures using a call to the CSNDDSG verb (Digital_Signature_Generate) naming the retained private key generated in the prior step and employing either the **X9.31** or the **ISO-9796** hash-formatting methods.
4. Support logon operations via the CSUALCT verb (Logon_Control) on the processing threads which invoke key generation and digital signature generation.
5. Output the public key value and the key-label used to access the retained private key.
6. Ensure that the application can operate properly in an environment with the access-control restrictions listed in Table 6-2 on page 6-19.

Policy Requirements

These policy requirements are focused on the Coprocessor and must be augmented through consideration of the host-system operating system and digital signing application, the physical environment, and so forth, which are necessarily unique to each installation.

Policy requirements address subjects including:

- Required tasks
- Personnel assignments.

⁷ Employ a skeleton key-token (from the PKA_Key_Generate verb) with the **SIG-ONLY** (signature only) attribute.

⁸ The application may permit the introduction of a nonce to be included in the self-signed public-key certificate returned from the PKA_Key_Generate verb. Note that this does not guarantee that the private key exists in a Coprocessor, but does provide additional assurance and integrity on the public-key value.

⁹ Uses the **RETAIN** rule-array keyword on the PKA_Key_Generate verb.

¹⁰ The *regeneration_data_length* variable in the CSNDPKG (PKA_Key_Generate) call must be valued to zero.

¹¹ The key label also must not exist in the host system key-storage.

Required Tasks: The organization must establish a complete set of rules (policy) governing the selection, installation, commissioning, operation, decommissioning, and auditing of the proposed digital-signing server. With reference to the Coprocessor, consider these items:

1. Reference Appendix G, "Threat Considerations for a Digital-Signing Server" to serve as a cross-check on your decisions.
2. Assign individuals to roles as discussed in "Personnel Assignments."
3. Identify, procure, install, and test a host-system computer paying attention to the security requirements which the operating system should support. You need to limit the exposure from rogue processes which might:
 - Intercept passphrases
 - Modify the data to be signed after the data has been approved for signing
 - Substitute an alternative cryptographic process that could be subverted.
4. Install an IBM 4758 Model 002 or an IBM eServer pSeries feature code #4963 and Release 2.41 CCA software and prerequisite software as documented earlier in this manual.
 - The CCA Support Program software must be installed as indicated in Chapters 3, 4, and up through the section "How to Establish a Test Node" on page 5-3.
 - An auditor should use the CLU utility VA command to confirm that the code loaded in the Coprocessor is validated, has the correct release identifiers, and has "Seg" 2 and 3 owner identifiers valued to two (2). See Figure 4-1 on page 4-3 and "Validating the Coprocessor Segment Contents" on page 4-5.
5. Identify, procure, install, and test a host-system application program that conforms to your requirements and which can employ the Coprocessor within the constraints described under the "Application Requirements" on page 6-14.
6. Limit access to the environment of the Coprocessor.
7. Make maximum use of host-system security features.
8. Carry out the tasks detailed in "Operational Requirements and Plan" on page 6-16.
9. Provide for periodic audit review of the system and associated records.

Personnel Assignments: Related to the use of the Coprocessor, assign and train individuals to serve in these capacities:

Installer An individual or team who makes the initial installation of the hardware and software for the application environment.

Administrator An individual¹² responsible for the *Setup* phase of the installation including promoting the system to the *key generation* phase.

The individual may be granted authority and technical capability to terminate normal production, or such authority and capability may be granted to another person such as an *auditor*.

¹² Two or more administrators may be assigned to work together to ensure that the required tasks, and no more, are performed correctly and in sequence. An objective is that the administrators will not collude with each other.

b **Auditor** An individual responsible for confirming that the installation is in an
b appropriate condition for entering and continuing in production, and who
b can report problems to authorities independent from others involved with
b the system.

b The *auditor* may be granted authority and technical capability to
b terminate normal production, or such authority and capability may be
b granted to an additional *auditor*.

b **Key Generator** An individual responsible for the key-pair generation phase of
b operation. This individual may also be responsible for obtaining
b certification of the generated public key. (You may incorporate this
b responsibility in the role of either the *administrator* or the *signer*.)

b **Signer** An individual responsible for conditioning the system to use the
b digital-signing key.

Operational Requirements and Plan

b These operational requirements and plan steps are focused on the Coprocessor
b and must be augmented through consideration of the host-system operating system
b and digital signing application, the physical environment, and so forth, which are
b necessarily unique to each installation.

b The secure usage of the Coprocessor is dependent on the definition of roles in
b accordance with a specific usage policy. A *role* defines a set of permissible
b Coprocessor *commands*. To perform a CCA *verb* (service) requires the use of one
b or more commands. A command is enabled by a *permission* in the *active role*. If
b the caller of a verb is not logged on, the DEFAULT role is active. There is always
b a DEFAULT role. A user may log on by referencing his *profile* and presenting his
b passphrase. The user's profile identifies a role which expresses his permissions to
b enable commands and thus CCA verbs.

b The CCA Support Program provides considerable flexibility in the configuration of
b roles. Other suitable ways to set up a secure CCA system as a digital-signing
b server exist beyond the example described here. For more details of the concept
b of roles within the Coprocessor, refer to the introductory material in Chapter 2 of
b *IBM 4758 CCA Basic Services Reference and Guide* and Appendix A, "CCA
b Access-Control Commands" in this book.

b The plan should address these phases of activity related to the Coprocessor:

Checkout

b In this phase the hardware and software are installed and preliminary tests
b are performed, individuals are trained, and so forth. See Chapters 1
b through 5 in this manual.

Setup

b In this phase the specific set of access controls are established by the
b *administrator* as listed in Table 6-2 on page 6-19. Perform these steps:

- b 1. Begin this activity by reinstalling the CCA software using the CLU utility
b and the software file CNWxxxxx.CLU as described in Chapter 4, "Loading
b Software into the Coprocessor."
- b 2. The *auditor* must (re)confirm that the proper CCA Support Program
b software is installed and that the segment 2 and 3 segment identifiers
b are two (2). (Use the CLU utility. See Chapter 4, "Loading Software into
b the Coprocessor." The *auditor* may also wish to confirm that the

- b remainder of the CCA Support Program has been loaded from the IBM
b Web site, <http://www.ibm.com/security/cryptocards>.)
- b 3. Follow the procedure listed in "How to Establish a Test Node" on
b page 5-3 to condition the CCA application within the Coprocessor.
 - b 4. Create the specific roles as listed in Table 6-2 on page 6-19.
 - b 5. Create the profiles you will require to access the roles. Each individual
b should have his own profile. It is technically possible to have multiple
b individuals (and therefore profiles) able to access a single role. Your
b installation policy will dictate what profiles to establish.
 - b 6. Update the **DEFAULT** role with the permissions defined for the **OP-DEF**
b role. See Table 6-2 on page 6-19.
 - b 7. Delete profile(s) that point to the **SETUP** role. This prevents the addition
b or modification of roles and profiles.
 - b 8. The *auditor* should (re)verify:
 - b • The installed software including the code loaded in segments 2 and 3
b ensuring that the segment owner identifiers are valued to two (2).
 - b • The set of installed profiles and roles, and the permissions in each
b role. (Use the CNM utility or other trusted tool.)
 - b 9. Have the individual users modify the passphrase for their profile by
b signing on to the **SETUP-1** role. Then delete profile(s) that point to the
b **SETUP-1** role. Note that any user's passphrase can be changed when
b the Change User Profile Authentication Data command (X'0114') is
b enabled.
 - b 10. Have the *auditor*:
 - b • Confirm that the profiles which invoke the **SETUP-1** role have been
b deleted
 - b • Obtain test patterns for the current and old master keys.

Key Generate

Have the *key-generator*:

- b 1. Generate the required public-private key(s). Note the key label which the
b application program uses to reference the generated key-pair.
- b 2. Then generate a random master key and set this master key. Repeat this
b process so that both the current and old master-key registers contain
b random, unknown master keys. (This prevents the use of any preexisting
b master-key-encrypted private keys. You can use the CNM Utility.)
- b 3. Have the *auditor* confirm:
 - b • That the master-key test patterns obtained earlier will not verify either
b the current or the old master key. (You can use the CNM Utility.)
 - b • The existence of the key label of the retained private key in the
b Coprocessor through the use of the Retained_Key_List verb. (You
b can use the CNM Utility.)
 - b • Observe the generation and verification of a test message. The public
b key may be used in another system to validate both the test digital
b signature and the self-signature on the generated public key.

Table 6-2 (Page 2 of 2). Roles and Permissions for a Digital-Signing Server	
Roles	Permissions
KEY-GEN Key Generator Use this role to generate RSA key-pair(s) through the use of the PKA_Key_Generate verb.	X'001A', (MKP) Set Master Key X'001D', (KSI) Compute Verification Pattern X'0020', (MKP) Generate Random Master Key X'0230', (RKL) List Retained Keys X'0103', (PKG) PKA_Key_Generate X'0230', (RKL) List Retained Keys
SIGNER Use this role to create digital signatures through the use of the Digital_Signature_Generate verb.	X'001D', (DSG) Compute Verification Pattern X'0100', (DSG) Digital_Signature_Generate X'0101', (DSV) Digital_Signature_Verify X'0103', (PKG) PKA_Key_Generate X'0107', (OWH) One-Way Hash, SHA-1 X'0203', (RKD) Delete Retained Key† X'0230', (RKL) List Retained Keys
(...), the last three letters in the verb entry-point name † An optional command ‡ Possibly an auditor should be able to disable use of the cryptographic facility or a specific key. Enablement of these commands for the auditor is an application design issue. ◊ If a distinct role is used to generate RSA key-pairs, the Signer role would omit key generation.	

Alternatives to the Role Scheme: These variations to the role scheme can be considered:

Key Generation You can eliminate the role used specifically for key generation and grant that authority to any one of the **SIGNER**, **SETUP-1**, or **SETUP** roles.

An advantage of the distinct **KEY-GEN** role is an improved procedure for enabling the auditor to confirm the appropriateness of the generated key prior to productive use.

An advantage in permitting the *signer* to generate the key lies in the fact that the *signer* is responsible for use of the key.

Permitting the **SETUP** role to generate the key can reduce the number of roles. However, note that under the **SETUP** role, the access controls remain changeable. It may be advantageous to delay key generation until the access controls are rendered unchangeable.

Private Key Deletion The private key(s) can be deleted by zeroizing the segment 2 and 3 software. It is also possible to delete a retained private-key using the Retained_Key_Delete verb which uses the Delete Retained Key command (X'0203'). You should decide when and who will have the responsibility for key deletion (zeroization). As appropriate, enable the Delete Retained Key command in any of the defined roles.

Passphrase Changes You can eliminate the **SETUP-1** role if you have the user passphrases set using the **SETUP** role. The **SETUP-1** role enables an *auditor* to first check the roles prior to having the various users come and establish their passphrases.

1 Secured Code-Signing Node

1 IBM offers Toolkits for creating custom code that runs in the Coprocessor. When
1 development of the custom code is complete, the code can be digitally signed so
1 that Coprocessors will accept the code and report ownership under the identifier
1 assigned to the developer. Using the *Signer* and *Packager* utilities provided in the
1 Toolkit Release 2.41, the access controls for the “code-signing” CCA node can
1 enable control over the generation and use of the private keys.

1 Two private keys are used by the Signer utility, and a third key is used by the
1 Packager utility. The Signer utility can be used to generate all three keys using the
1 **CRUSIGNR KEYGEN 0** command as described in Chapter 5 of the *IBM 4758 PCI*
1 *Cryptographic Coprocessor Custom Software Developer's Guide*. The private keys
1 are encrypted by the CCA master keys. To backup the private keys, make
1 duplicate copies of the key files created by the Signer utility. Consider use of
1 master-key cloning to another Coprocessor so that any loss of the master key on
1 the primary signing node will still enable use of the key files on the backup node.
1 Before continuing with this material, you should understand the introductory
1 material in Chapter 2 of *IBM 4758 CCA Basic Services Reference and Guide* and
1 familiarize yourself with Appendix A, “CCA Access-Control Commands” in this
1 book.

1 Consider use of these roles:

1 **Initial Default** Use this or a similar default role to check out the Coprocessor
1 installation and CCA software. See “How to Establish a Test Node” on
1 page 5-3.

1 At the conclusion of the checkout, zeroize the node. See “How to Initialize
1 (Zeroize) the Node” on page 5-7.

1 **Setup** Use this role during node set up exclusive of master-key cloning, RSA
1 key-generation, and code-signing digital-signature generation. This role
1 permits modifications to the access-control system, therefore, use of the role
1 should be disabled prior to normal operations so as to lock the access-control
1 system. Delete all profiles that enable this role using
1 Access_Control_Maintenance with the **DEL-PROF** keyword, or that command
1 via the CNM utility.

1 **Auditor** Use this role prior to the start of normal operations, and later as required,
1 to confirm the access-control system settings.

1 The auditor should confirm that the roles and profiles within the Coprocessor
1 are appropriate.

1 The auditor might also be assigned the right(s) to zeroize the entire
1 cryptographic node.

1 **Administrator-1** An individual authorized to participate in master-key creation and
1 optionally in cloning of master keys.

1 **Administrator-2** An individual authorized to participate in activating master keys
1 and therefore the CCA node(s), and optionally in cloning of master keys.

1 **Key-Generator** An individual authorized to generate the code-signing keys. Also
1 generate the public-private key-pairs used in master-key cloning.

1 **Signer** An individual authorized to use the code-signing keys. If using master-key
1 cloning, also certifies the master-key cloning CSS and CSR keys.

Table 6-3 on page 6-22 describes possible roles and appropriate permissions.

<i>Table 6-3 (Page 1 of 2). Roles and Permissions for a Simple CA Case</i>	
Roles	Permissions
<p>Initial Default</p> <p>This role is used to verify reasonable operation of the installed hardware and software. Once initial checkout is complete, this role is replaced by the Operational Default role and the other roles described next.</p>	<p>All</p> <p>(Use the "Enable All" button in the CNM utility.)</p>
<p>Setup</p> <p>This role can be used during initial application testing and establishment of test roles. Once proper operation is confirmed, the profile(s) which activates this role should be deleted because they can be used to alter and extend the access-control regime.</p>	<p>X'001A', (MKP) Set Master Key X'001D', (KSI) Compute Verification Pattern X'0020', (MKP) Generate Random Master Key X'0101', (DSV) Digital_Signature_Verify† X'0107', (OWH) One-Way Hash, SHA-1† X'010F', (CFC) Reset Intrusion Latch X'0110', (CFC) Cryptographic_Facility_Control X'0111', (CFC) Reinitialize Device X'0112', (ACI) Initialize Access-Control System X'0113', (ACI) Change User Profile Expiration Date X'0114', (ACI) Change User Profile Authentication Data X'0115', (ACI) Reset User Profile Logon-Attempt-Failure Count X'0116', (ACM) Read Public Access-Control Information X'0117', (ACM) Delete User Profile X'0118', (ACM) Delete Role X'0119', (CFC) Load Function-Control Vector X'011A', (CFC) Clear Function-Control Vector† X'011C', (CFC) Set EID† X'011D', (CFC) Initialize Master Key Cloning†</p>
<p>Operational Default</p> <p>(Replaces Initial Default.)</p> <p>This role is in effect if any call is made to the CCA Coprocessor function from a caller who has not successfully logged on to the Coprocessor.</p>	<p>X'0116', (ACM) Read Public Access-Control Information</p>
<p>Auditor</p> <p>This role is used to query the access-controls setup and to confirm that the profile(s) that could activate the setup role have been deleted prior to sanctioning start-up of normal operations.</p>	<p>X'0107', (OWH) One-Way Hash, SHA-1† X'0111', (CFC) Reinitialize Device‡ X'0116', (ACM) Read Public Access-Control Information X'0117', (ACM) Delete User Profile‡</p>
<p>(...), the last three letters in the verb entry-point name</p> <p>† An optional command</p> <p>‡ Possibly an auditor should be able to disable use of the cryptographic facility or a specific key. Enablement of these commands for the auditor is an installation-practices issue.</p>	

Table 6-3 (Page 2 of 2). Roles and Permissions for a Simple CA Case

Roles	Permissions
<p>Administrator-1</p> <p>Establishes master key and works with first of two-part key operations.</p>	<p>X'0020', (MKP) Generate Random Master Key X'0022', (MKP) Clear New Master Key X'008E', (RNG) Generate Key (and random number) X'0101', (DSG) Digital Signature Verify X'0107', (OWH) One-Way Hash, SHA-1 X'0200', (PKH) Register Public Key Hash X'0211', (MKD) Clone Share 1 Generate X'0221', (MKD) Clone Share 1 Install</p>
<p>Administrator-2</p> <p>Activates master key and work with second of two-part key operations.</p>	<p>X'001A', (MKP) Set Master Key X'0033', (MKP) Clear Old Master Key X'008E', (RNG) Generate Key (and random number) X'0201', (PKR) Register Public Key X'0212', (MKD) Clone Share 2 Generate X'0222', (MKD) Clone Share 2 Install X'0107', (OWH) One-Way Hash, SHA-1</p>
<p>Key Generator</p> <p>Generates the code-signing RSA keys. Also generate the public-private key-pairs used in master-key cloning.</p>	<p>X'008E', (RNG) Generate Key (and random number) X'0103', (PKG) PKA_Key_Generate X'0107', (OWH) One-Way Hash, SHA-1 X'0204', (PKG) PKA Clone Key Generate†</p>
<p>Signer</p> <p>Uses the Signer and Packager utilities for signing code. If using master-key cloning, also certifies the master-key cloning CSS and CSR keys.</p>	<p>X'008E', (RNG) Generate Key (and random number) X'0100', (DSG) Digital_Signature_Generate X'0101', (DSV) Digital_Signature_Verify X'0107', (OWH) One-Way Hash, SHA-1</p>
<p>(...), the last three letters in the verb entry-point name</p> <p>† An optional command</p> <p>‡ Possibly an auditor should be able to disable use of the cryptographic facility or a specific key. Enablement of these commands for the auditor is an installation-practices issue.</p>	

Chapter 7. Building Applications to Use with the CCA API

This chapter includes the following:

- An overview of the way in which applications obtain service from the Common Cryptographic Architecture (CCA) application program interface (API)
- The procedure for calling a CCA verb in the C programming language
- The procedure for compiling applications and linking them to the CCA API
- A sample routine written in the C programming language
- Enhancing throughput with CCA and the 4758 Models 002 and 023.

Source code for the sample routine is shipped with the software. You can use the sample included to test the Coprocessor and the support program.

Note: The file locations referenced in this chapter are the default directory paths.

Overview

Application and utility programs issue service requests to the PCI Cryptographic Coprocessor by calling the CCA API verbs¹. The Windows NT/2000 environments link CCA API requests to their dynamic link library (DLL) code, and AIX links requests to its shared library code. The operating system code in turn calls the Coprocessor physical device driver (PDD). The hardware and software accessed through the API are themselves an integrated subsystem.

Verb calls are written in the standard syntax of the C programming language, and include an `entry_point_name`, verb parameters, and the variables for those parameters. The same `entry_point_name`, parameters, and variables are used in AIX, Windows NT and Windows 2000 environments, so code can be ported between them with minimal change.

For a detailed listing of the verbs, variables, and parameters you can use when programming for the CCA API, refer to the *IBM 4758 CCA Basic Services Reference and Guide*.

How to Call Verbs in C Program Syntax

In every operating system environment, you can code verb calls using standard C programming language syntax.

Function call prototypes for all CCA API verbs are contained in the include file. The files and their default distribution locations are:

AIX	/usr/include/csufincl.h
Windows NT/2000	\Program Files\IBM\4758\include\csunincl.h

To include these verb declarations, use the following compiler directive in your program:

¹ The term “verb” implies an action that an application program can initiate. Some systems and publications use the term “callable service.”

```
AIX                #include <csufincl.h>
Windows NT/2000   #include "csunincl.h"
```

When you issue a call to a CCA API verb, code the verb entry_point_name in uppercase characters. Separate the parameter identifiers with commas and enclose them in parentheses. End the call with a semicolon character. For example:

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

Note: The third and fourth parameters of a CCA call, *exit_data_length* and *exit_data*, are not currently supported by the support program. Although it is permissible to code null address pointers for these parameters, it is recommended that you specify a long integer valued to zero with the *exit_data_length* parameter.

How to Compile and Link Application Programs

The support program includes the C Language source code and the make file for a sample program. The file and its default distribution location is:

```
AIX                /usr/lpp/csaf/samples/c
Windows NT/2000   \Program Files\IBM\4758\samples
```

To compile application programs which use CCA, you can use the IBM VisualAge C compiler tools, or similar tools from other vendors.

Link the compiled program to the CCA library. The library and its default distribution location is:

```
AIX                /usr/lib/libcsufsapi.a
Windows NT/2000   \Program Files\IBM\4758\lib\csunsapi.lib
```

Compiling Applications for AIX

When compiling your applications for AIX, use the *_r* suffixed version of the compiler. The *_r* suffixed compiler supports multi-threaded operation. For example, *xlcr*.

Sample Routine

To illustrate the practical application of CCA verb calls, this section describes the sample routine included with the support program. For reference, a hard copy of the sample routine is shown in Figure 7-1 on page 7-4. (There is also a sample program on the product Web site. That sample program can help you understand the performance of the CCA implementation.)

The sample routine generates a message authentication code (MAC) on a text string and then verifies the MAC. To effect this, the routine:

1. Calls the Key_Generate (CSNBKGN) verb to create a MAC/MACVER key pair.
2. Calls the MAC_Generate (CSNBMGN) verb to generate a MAC on a text string with the MAC key.
3. Calls the MAC_Verify (CSNBMVR) verb to verify the text string MAC with the MACVER key.

As you review the sample routine shown in Figure 7-1 on page 7-4, refer to the *IBM 4758 CCA Basic Services Reference and Guide* for descriptions of the called verbs and their parameters. These verbs are listed in Table 7-1.

Verb	Entry_Point_Name
Key_Generate	CSNBKGN
MAC_Generate	CSNBMGN
MAC_Verify	CSNBMVR

```

/*****/
/* */
/* Module Name: mac.c */
/* */
/* DESCRIPTIVE NAME: Cryptographic Coprocessor Support Program */
/* C language source code example */
/* */
/*-----*/
/* */
/* Licensed Materials - Property of IBM */
/* */
/* (C) Copyright IBM Corp. 1997-2001 All Rights Reserved */
/* */
/* US Government Users Restricted Rights - Use duplication or */
/* disclosure restricted by GSA ADP Schedule Contract with IBM Corp. */
/* */
/*-----*/
/* */
/* NOTICE TO USERS OF THE SOURCE CODE EXAMPLES */
/* */
/* The source code examples provided by IBM are only intended to */
/* assist in the development of a working software program. The */
/* source code examples do not function as written: additional */
/* code is required. In addition, the source code examples may */
/* not compile and/or bind successfully as written. */
/* */
/* International Business Machines Corporation provides the source */
/* code examples, both individually and as one or more groups, */
/* "as is" without warranty of any kind, either expressed or */
/* implied, including, but not limited to the implied warranties of */
/* merchantability and fitness for a particular purpose. The entire */
/* risk as to the quality and performance of the source code */
/* examples, both individually and as one or more groups, is with */
/* you. Should any part of the source code examples prove defective, */
/* you (and not IBM or an authorized dealer) assume the entire cost */
/* of all necessary servicing, repair or correction. */
/* */
/* IBM does not warrant that the contents of the source code */
/* examples, whether individually or as one or more groups, will */
/* meet your requirements or that the source code examples are */
/* error-free. */
/* */
/* IBM may make improvements and/or changes in the source code */
/* examples at any time. */
/* */
/* Changes may be made periodically to the information in the */
/* source code examples; these changes may be reported, for the */
/* sample code included herein, in new editions of the examples. */
/* */
/* References in the source code examples to IBM products, programs, */
/* or services do not imply that IBM intends to make these */
/* available in all countries in which IBM operates. Any reference */
/* to the IBM licensed program in the source code examples is not */
/* intended to state or imply that IBM's licensed program must be */
/* used. Any functionally equivalent program may be used. */
/* */

```

Figure 7-1 (Part 1 of 5). Syntax, Sample Routine

```

/*-----*/
/*
/* This example program:
/*
/* 1) Calls the Key_Generate verb (CSNBKGN) to create a MAC (message
/* authentication code) key token and a MACVER key token.
/*
/*
/* 2) Calls the MAC_Generate verb (CSNBMGN) using the MAC key token
/* from step 1 to generate a MAC on the supplied text string
/* (INPUT_TEXT).
/*
/*
/* 3) Calls the MAC_Verify verb (CSNBMVR) to verify the MAC for the
/* same text string, using the MACVER key token created in
/* step 1.
/*
/*
/*-----*/
#include <stdio.h>
#include <string.h>

#ifdef _AIX
#include <csufincl.h>
#elif __WINDOWS__
#include "csunincl.h"
#else
#include "csueincl.h"
#endif

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

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

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

```

Figure 7-1 (Part 2 of 5). Syntax, Sample Routine

```

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

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

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

```

Figure 7-1 (Part 3 of 5). Syntax, Sample Routine

```

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

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

/* Check the return/reason codes. Terminate if there is an error. */
if (return_code != 0 || reason_code != 0) {
    printf ("MAC Generate Failed: ");        /* Print failing verb */
    printf ("return_code = %ld, ", return_code); /* Print return code */
    printf ("reason_code = %ld.\n", reason_code); /* Print reason code */
    return;
}
else {
    printf ("MAC_Generate successful.\n");
    printf ("MAC_value = %s\n", mac_value); /* Print MAC value (HEX-9) */
}

```

Figure 7-1 (Part 4 of 5). Syntax, Sample Routine

```

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

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

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

```

Figure 7-1 (Part 5 of 5). Syntax, Sample Routine

Enhancing Throughput with CCA and the 4758 Models 002 and 023

When you use the CCA API, the characteristics of your host application program will affect performance and throughput of the 4758-002 and 4758-023. There are two areas you should understand in order to evaluate performance and design your application to obtain the best performance from the 4758 Coprocessor. One is multi-threading and multi-processing and the other is caching DES and RSA keys.

Multi-threading and Multi-processing

The CCA application running inside the 4758 models 002 and 023 can process several CCA requests simultaneously. The Coprocessor contains several independent hardware elements, including the RSA engine, DES engine, CPU, random-number generator, and PCI communications interface. These elements can all be working at the same time, processing parts of different CCA verbs. By working on several verbs at the same time, the 4758 can keep all of its hardware elements busy, maximizing the overall system throughput.

In order to take advantage of this capability, your host system must send multiple CCA requests to the 4758 without waiting for each one to finish before sending the next one. The best way to accomplish this is to design a multi-threaded host application program, in which each thread can independently send CCA requests to the 4758. For example, a Web server can start a new thread for each request it receives over the network. Each of these threads will send the required cryptographic requests to the 4758, independent of what the other threads are

doing. By doing this, you guarantee that the 4758 is not under utilized. Another option is to have several independent host application programs all using the 4758 at the same time.

Caching DES and RSA Keys

The CCA software for the 4758 models 002 and 023 keeps copies of recently used DES and RSA keys in caches, inside the secure module. The keys are stored in a form that has been decrypted and validated, and is ready for use. If the same key is reused in a later CCA request, the 4758 can use the cached copy and avoid the overhead associated with decrypting and validating the key token. In addition, for retained RSA keys, the cache eliminates the overhead of retrieving the key from the internal flash EPROM memory.

As a result, applications that reuse a common set of keys can run much faster than those which use different keys for each transaction. Most common applications use a common set of DES keys and RSA private keys, and the caching is very effective in improving throughput. RSA public keys, which have very little processing overhead, are not cached.

Appendix A. CCA Access-Control Commands

The table in this appendix lists the CCA access-control commands (“control points”) supported by the CCA Cryptographic Coprocessor Support Program. The role to which a user is assigned determines the commands available to that user.

Important: By default, you should disable commands. Do not enable a command unless you know why you are enabling it.

The table includes the following columns:

Offset	The hexadecimal offset for the command; offsets between X'0000' and X'FFFF' not listed in this table are reserved.
Command Name	The name of the command required by the following verbs.
Verb Name	The names of the verbs that require that command to be enabled; for example, the Encipher (CSNBENC) verb will fail without permission to use the Encipher command.
Entry	The entry_point_name of the verb.
Usage	Usage recommendations for the command. The abbreviations in this column are explained at the bottom of the page.

For information about the verbs and the functions they call, refer to the *IBM 4758 CCA Basic Services Reference and Guide*.

Table A-1 (Page 1 of 4). Supported CCA Commands

Code	Command Name	Verb Name	Entry	Usage
X'000E'	Encipher	Encipher	CSNBENC	O
X'000F'	Decipher	Decipher	CSNBDEC	O
X'0010'	Generate MAC	MAC_Generate	CSNBMGN	O
X'0011'	Verify MAC	MAC_Verify	CSNBMVR	O
X'0012'	Reencipher to Master Key	Key_Import	CSNBKIM	O
X'0013'	Reencipher from Master Key	Key_Export	CSNBKEX	O
X'0018'	Load First Master Key Part	Master_Key_Process†	SNBMKP	SC, SEL
X'0019'	Combine Master Key Parts	Master_Key_Process†	CSNBMKP	SC, SEL
X'001A'	Set Master Key	Master_Key_Process†	CSNBMKP	SC, SEL
X'001B'	Load First Key Part	Key_Part_Import†	CSNBKPI	SC, SEL
X'001C'	Combine Key Parts	Key_Part_Import†	CSNBKPI	SC, SEL
X'001D'	Compute Verification Pattern	Key_Test Key_Storage_Initialization DES_Key_Record_Create DES_Key_Record_Delete DES_Key_Record_List DES_Key_Record_Read DES_Key_Record_Write PKA_Key_Record_Create PKA_Key_Record_Delete PKA_Key_Record_List PKA_Key_Record_Read PKA_Key_Record_Write	CSNBKYT CSNBKSI CSNBKRC CSNBKRD CSNBKRL CSNBKRR CSNBKRW CSNDKRC CSNDKRD CSNDKRL CSNDKRR CSNDKRW	R
X'001F'	Translate Key	Key_Translate	CSNBKTR	O
X'0020'	Generate Random Master Key	Master_Key_Process†	CSNBMKP	O, SEL
X'0032'	Clear New Master Key Register	Master_Key_Process†	CSNBMKP	O, SUP
X'0033'	Clear Old Master Key Register	Master_Key_Process†	CSNBMKP	O, SUP
X'0040'	Generate Diversified Key (CLR8-ENC)	Diversified_Key_Generate‡	CSNBDBG	O, SEL
X'0041'	Generate Diversified Key (TDES-ENC)	Diversified_Key_Generate‡	CSNBDBG	O, SEL
X'0042'	Generate Diversified Key (TDES-DEC)	Diversified_Key_Generate‡	CSNBDBG	O, SEL
X'0043'	Generate Diversified Key (SESS-XOR)	Diversified_Key_Generate‡	CSNBDBG	O, SEL
X'0044'	Enable DKG Single Length Keys and Equal Halves for TDES-ENC, TDES-DEC	Diversified_Key_Generate‡	CSNBDBG	SC, SEL
X'0053'	Load First Asymmetric Master Key Part	Master_Key_Process†	CSNBMKP	SC, SEL
X'0054'	Combine PKA Master Key Parts	Master_Key_Process†	CSNBMKP	SC, SEL
X'0057'	Set Asymmetric Master Key	Master_Key_Process†	CSNBMKP	SC, SEL
X'0060'	Clear New Asymmetric Master Key Buffer	Master_Key_Process†	CSNBMKP	SC, SEL
X'0061'	Clear Old Asymmetric Master Key Buffer	Master_Key_Process†	CSNBMKP	SC, SEL
X'008A'	Generate MDC	Generate_Modification_Detection_Code	CSNBMDG	R
X'008C'	Generate Key Set	Key_Generate‡	CSNBKGN	O
X'008E'	Generate Key	Key_Generate‡ Random_Number_Generate	CSNBKGN CSNBRNG	R
X'0090'	Reencipher to Current Master Key	Key_Token_Change	CSNBKTC	R
<p>The following codes are used in this table:</p> <p>ID Initial default. O Usage of this command is optional; enable it as required for authorized usage. R Enabling this command is recommended. NR Enabling this command is not recommended. NRP Enabling this command is not recommended for production. SC Usage of this command requires special consideration. SEL Usage of this command is normally restricted to one or more selected roles. SUP This command is normally restricted to one or more supervisory roles.</p> <p>† This verb performs more than one function, as determined by the keyword in the <i>rule_array</i> parameter of the verb call. Not all functions of the verb require the command in this row. ‡ This verb does not always require the command in this row. Use as determined by the control vector for the key and the action being performed.</p>				

Table A-1 (Page 2 of 4). Supported CCA Commands

Code	Command Name	Verb Name	Entry	Usage
X'00A0'	Generate Clear 3624 PIN	Clear_PIN_Generate	CSNBPGN	O
X'00A4'	Generate Clear 3624 PIN Offset	Clear_PIN_Generate_Alternate†	CSNBCPA	O
X'00AB'	Verify Encrypted 3624 PIN	Encrypted_PIN_Verify†	CSNBPVR	O
X'00AC'	Verify Encrypted German Bank Pool PIN	Encrypted_PIN_Verify†	CSNBPVR	O
X'00AD'	Verify Encrypted VISA PVV	Encrypted_PIN_Verify†	CSNBPVR	O
X'00AE'	Verify Encrypted Interbank PIN	Encrypted_PIN_Verify†	CSNBPVR	O
X'00AF'	Format and Encrypt PIN	Clear_PIN_Encrypt	CSNBCPE	O
X'00B0'	Generate Formatted and Encrypted 3624 PIN	Encrypted_PIN_Generate†	CSNBEPG	O
X'00B1'	Generate Formatted and Encrypted German Bank Pool PIN	Encrypted_PIN_Generate†	CSNBEPG	O
X'00B2'	Generate Formatted and Encrypted Interbank PIN	Encrypted_PIN_Generate†	CSNBEPG	O
X'00B3'	Translate PIN with No Format-Control to No Format-Control	Encrypted_PIN_Translate†	CSNBPTR	O
X'00B7'	Reformat PIN with No Format-Control to No Format-Control	Encrypted_PIN_Translate†	CSNBPTR	O
X'00BB'	Generate Clear VISA PVV Alternate	Clear_PIN_Generate_Alternate†	CSNBCPA	O
X'00C3'	Encipher Under Master Key	Clear_Key_Import Multiple_Clear_Key_Import	CSNBCKI CSNBCKM	SC
X'00CD'	Lower Export Authority	Prohibit_Export	CSNBPEX	O
X'00D6'	Translate Control Vector	Translate_Control_Vector	CSNBCVT	SC
X'00D7'	Generate Key Set Extended	Key_Generate‡	CSNBKGN	SC, SUP
X'00DA'	Encipher/Decipher Cryptovvariable	Cryptographic_Variable_Encipher	CSNBCVE	NRP, O, SUP
X'00DB'	Replicate Key	Key_Generate‡	CSNBKGN	NR, SC
X'00DF'	Generate CVV	CVV_Generate	CSNBCEG	O
X'00E0'	Verify CVV	CVV_Verify	CSNBCEV	O
X'00E1'	Unique Key Per Transaction, ANSI X9.24	Encrypted_PIN_Translate†	CSNBPTR	O
X'0100'	PKA96 Digital Signature Generate	Digital_Signature_Generate	CSNDDSG	O, SC
X'0101'	PKA96 Digital Signature Verify	Digital_Signature_Verify	CSNDDSV	O
X'0102'	PKA96 Key Token Change	PKA_Key_Token_Change	CSNDKTC	O
X'0103'	PKA96 PKA Key Generate	PKA_Key_Generate†	CSNDPKG	O, SUP
X'0104'	PKA96 PKA Key Import	PKA_Key_Import	CSNDPKI	O, SUP
X'0105'	Symmetric Key Export PKCS-1.2/OAEP	PKA_Symmetric_Key_Export	CSNDSYX	SC
X'0106'	PKA Symmetric Key Import PKCS-1.2/OAEP	PKA_Symmetric_Key_Import†	CSNDSYI	O
X'0107'	One-Way Hash, SHA-1	One_Way_Hash	CSNBOWH	R
X'0109'	Data Key Import	Data_Key_Import	CSNBDKM	O
X'010A'	Data Key Export	Data_Key_Export	CSNBDKX	O
X'010B'	Compose SET Block	SET_Block_Compose	CSNDSBC	O
X'010C'	Decompose SET Block	SET_Block_Decompose	CSNDSBD	O

The following codes are used in this table:

- ID** Initial default.
- O** Usage of this command is optional; enable it as required for authorized usage.
- R** Enabling this command is recommended.
- NR** Enabling this command is **not** recommended.
- NRP** Enabling this command is **not** recommended for production.
- SC** Usage of this command requires special consideration.
- SEL** Usage of this command is normally restricted to one or more selected roles.
- SUP** This command is normally restricted to one or more supervisory roles.

- † This verb performs more than one function, as determined by the keyword in the *rule_array* parameter of the verb call. Not all functions of the verb require the command in this row.
- ‡ This verb does not always require the command in this row. Use as determined by the control vector for the key and the action being performed.

Table A-1 (Page 3 of 4). Supported CCA Commands

Code	Command Name	Verb Name	Entry	Usage
X'010D'	PKA92 Symmetric Key Generate	PKA_Symmetric_Key_Generate†	CSNDSYG	SC
X'010E'	NL-EPP-5 Symmetric Key Generate	PKA_Symmetric_Key_Generate†	CSNDSYG	O
X'010F'	Reset Intrusion Latch	Cryptographic_Facility_Control†	CSUACFC	SUP
X'0110'	Set Clock	Cryptographic_Facility_Control†	CSUACFC	ID, SUP
X'0111'	Reinitialize Device	Cryptographic_Facility_Control†	CSUACFC	ID, SUP
X'0112'	Initialize Access-Control System	Access_Control_Initialization†	CSUAACI	ID, NRP, SUP
X'0113'	Change User Profile Expiration Date	Access_Control_Initialization†	CSUAACI	ID, SUP
X'0114'	Change User Profile Authentication Data	Access_Control_Initialization†	CSUAACI	ID, NRP, SUP
X'0115'	Reset User Profile Logon-Attempt-Failure Count	Access_Control_Initialization†	CSUAACI	ID, SUP
X'0116'	Read Public Access-Control Information	Access_Control_Maintenance†	CSUAACM	O, ID
X'0117'	Delete User Profile	Access_Control_Maintenance†	CSUAACM	ID, SUP
X'0118'	Delete Role	Access_Control_Maintenance†	CSUAACM	ID, SUP
X'0119'	Load Function-Control Vector	Cryptographic_Facility_Control†	CSUACFC	ID, NRP, SUP
X'011A'	Clear Function-Control Vector	Cryptographic_Facility_Control†	CSUACFC	NR, ID
X'011B'	Force User Logoff	Logon_Control†	CSUALCT	O, SUP
X'011C'	Set EID	Cryptographic_Facility_Control†	CSUACFC	O, SUP
X'011D'	Initialize Master Key Cloning	Cryptographic_Facility_Control†	CSUACFC	O, SUP
X'011E'	RSA Encipher Clear Key	PKA_Key_Encipher	CSNDPKE	O, SEL
X'011F'	RSA Decipher Clear Key	PKA_Key_Decipher	CSNDPKD	SC, SEL
X'0120'	Generate Random Asymmetric Master Key	Master_Key_Process†	CSNBMKP	SC, SEL
X'0121'	SET PIN Encrypt with IPINENC	SET_Block_Decompose†	CSNBSBD	O
X'0122'	SET PIN Encrypt with OPINENC	SET_Block_Decompose†	CSNBSBD	O
X'0200'	PKA Register Public Key Hash	PKA_Public_Key_Hash_Register	CSNDPKH	O
X'0201'	PKA Public Key Register with Cloning	PKA_Public_Key_Register†	CSNDPKR	O, SEL
X'0202'	PKA Public Key Register	PKA_Public_Key_Register†	CSNDPKR	O, SEL
X'0203'	Delete Retained Key	Retained_Key_Delete	CSNDRKD	O, SEL
X'0204'	PKA Clone Key Generate	PKA_Key_Generate†	CSNDPKG	O, SUP
X'0205'	PKA Clear Key Generate	PKA_Key_Generate†	CSNDPKG	O, SUP
X'0211' through X'021F'	Clone-info (Share) Obtain	Master_Key_Distribution†	CSNBMKD	O, SUP
X'0221' through X'022F'	Clone-info (Share) Install	Master_Key_Distribution†	CSNBMKD	O, SUP
X'0230'	List Retained Key	Retained_Key_List	CSNDRKL	O
X'0231'	Generate Clear NL-PIN-1 Offset	Clear_PIN_Generate_Alternate†	CSNBCPA	O
X'0232'	Verify Encrypted NL-PIN-1	Encrypted_PIN_Verify†	CSNBPVR	O
X'0235'	PKA92 Symmetric Key Import	PKA_Symmetric_Key_Import†	CSNDSYI	O

The following codes are used in this table:

- ID** Initial default.
- O** Usage of this command is optional; enable it as required for authorized usage.
- R** Enabling this command is recommended.
- NR** Enabling this command is **not** recommended.
- NRP** Enabling this command is **not** recommended for production.
- SC** Usage of this command requires special consideration.
- SEL** Usage of this command is normally restricted to one or more selected roles.
- SUP** This command is normally restricted to one or more supervisory roles.

- † This verb performs more than one function, as determined by the keyword in the *rule_array* parameter of the verb call. Not all functions of the verb require the command in this row.
- ‡ This verb does not always require the command in this row. Use as determined by the control vector for the key and the action being performed.

Table A-1 (Page 4 of 4). Supported CCA Commands

Code	Command Name	Verb Name	Entry	Usage
X'0236'	PKA92 Symmetric Key Import with PIN Keys	PKA_Symmetric_Key_Import†	CSNDSYI	O
X'023C'	ZERO-PAD Symmetric Key Generate	PKA_Symmetric_Key_Generate†	CSNDSYG	O
X'023D'	ZERO-PAD Symmetric Key Import	PKA_Symmetric_Key_Import†	CSNDSYI	O, SC
X'023E'	ZERO-PAD Symmetric Key Export	PKA_Symmetric_Key_Export†	CSNDSYX	O, SC
X'023F'	Symmetric Key Generate PKCS-1.2/OAEP	PKA_Symmetric_Key_Generate†	CSNDSYG	O, SC
1 X'0276'	Unrestrict Reencipher from Master Key	Key_Export	CSNBKEX	O, SC
1 X'0277'	Unrestrict Data Key Export	Data_Key_Export	CSNBKX	O, SC
1 X'0278'	Add Key Part	Key_Part_Import†	CSNBKPI	SC, SEL
1 X'0279'	Complete Key Part	Key_Part_Import†	CSNBKPI	SC, SEL
1 X'027A'	Unrestrict Combine Key Parts	Key_Part_Import	CSNBKPI	O, SC
1 X'027B'	Unrestrict Reencipher to Master Key	Key_Import	CSNBKIM	O, SC
1 X'027C'	Unrestrict Data Key Import	Data_Key_Import	CSNBKIM	O, SC
X'0290'	Generate Diversified Key (DALL with DKYGENKY Key Type)	Diversified_Key_Generate†	CSNDDKG	O, SC
X'0291'	Generate CSC-5, 4 and 3 Values	Transaction_Validate†	CSNBTRV	O, SEL
X'0292'	Verify CSC-3 Values	Transaction_Validate†	CSNBTRV	O
X'0293'	Verify CSC-4 Values	Transaction_Validate†	CSNBTRV	O
X'0294'	Verify CSC-5 Values	Transaction_Validate†	CSNBTRV	O
<p>The following codes are used in this table:</p> <p>ID Initial default.</p> <p>O Usage of this command is optional; enable it as required for authorized usage.</p> <p>R Enabling this command is recommended.</p> <p>NR Enabling this command is not recommended.</p> <p>NRP Enabling this command is not recommended for production.</p> <p>SC Usage of this command requires special consideration.</p> <p>SEL Usage of this command is normally restricted to one or more selected roles.</p> <p>SUP This command is normally restricted to one or more supervisory roles.</p> <p>† This verb performs more than one function, as determined by the keyword in the <i>rule_array</i> parameter of the verb call. Not all functions of the verb require the command in this row.</p> <p>‡ This verb does not always require the command in this row. Use as determined by the control vector for the key and the action being performed.</p>				

Appendix B. Initial DEFAULT-Role Commands

This appendix describes the characteristics of the DEFAULT role after the Coprocessor is initialized and when no other access-control data exists:

- The role ID is DEFAULT
- The required authentication strength is zero
- It is valid at all times of the day and on all days of the week
- The only functions permitted are those necessary to load access-control data.

Important

The cryptographic node is not secure when unauthenticated users can load access-control data using the DEFAULT role. Restrict these commands to selected supervisory roles.

Table B-1 lists the access-control commands enabled in the DEFAULT role when the CCA software is initially loaded and when the CCA node is initialized.

Code	Command Name
X'0107'	One-Way Hash
X'0110'	Set Clock
X'0111'	Reinitialize Device
X'0112'	Initialize Access-Control System
X'0113'	Change User Profile Expiration Date
X'0114'	Change User Profile Authentication Data (Passphrase)
X'0115'	Reset User Profile Logon-Attempt-Failure Count
X'0116'	Read Public Access-Control Information
X'0117'	Delete User Profile
X'0118'	Delete Role
X'0119'	Load Function Control Vector
X'011A'	Clear Function Control Vector

Appendix C. Machine-Readable-Log Contents

The CLU utility creates two log files, one intended for reading and the other for possible input to a program. This latter log file, the machine-readable log or MRL file, contains the binary outputs from the Coprocessor in response to various commands input to the Coprocessor.

Detailed information about the contents of the MRL is available from IBM 4758 Development. Contact IBM 4758 Development through use of the Support form on the IBM 4758 Web site.

Appendix D. Migration Considerations, Version 1 to 2

Version 1 of the CCA Support Program supports the IBM 4758 Models 001 and 013, while Version 2 of the CCA Support Program supports the IBM 4758 Models 002 and 023. Version 2 support is designed with greater consistency to the CCA implementation now available on S/390 through the Integrated Cryptographic Support Feature of OS/390. Certain of the design-point changes between Versions 1 and 2 are likely to cause some recoding of your application programs as you move to Version 2.

For an overview of the changes between the two Versions, see "Revision History" in the introductory material at the front of the *IBM 4758 CCA Basic Services Reference and Guide* manual.

Separate master key registers for both symmetric and asymmetric key encryption are provided. It is possible to independently operate on the register sets. Or, you can operate concurrently on the register sets in which case the results are comparable to the single set of registers used in the Version 1 CCA implementation. At any point if you operate only on one set of registers, all subsequent operations should operate independently on the two sets of master key registers.

The master key verification pattern used in the Version 2 implementation is eight bytes in length versus two bytes in the prior support. Also, the information that represents an RSA private key in CRT form is changed. While Version 2 will accept both Version 1 and Version 2 key tokens, it will only output key tokens in the Version 2 format. To the extent that your application is sensitive to the format of key tokens, some recoding may be required.

Routines that generate RSA-CRT keys will need to modify the keyword used in the PKA_Key_Generate verb. Use the new keyword **RSA-CRT**.

The Key-Token_Build for DES keys moves the information related to the master key verification pattern. Application programming recoding is required.

Most other changes are related to additional capabilities available with Version 2.

All applications that are functional with Version 1 must be thoroughly retested to confirm proper operation with Version 2 support.

Appendix E. Device Driver Error Codes

Each time that the Coprocessor is reset, and the reset is not caused by a fault or tamper event, the Coprocessor runs through "Miniboot," its power-on self-test (POST), code-loading, and status routines. During this process the Coprocessor attempts to coordinate with a host-system device driver. Coprocessor resets can occur because of power-on, a reset command sent from the device driver, or because of Coprocessor internal activity such as completion of code updates.

The Coprocessor can also reset if the Coprocessor's fault or tamper detection circuitry reset the Coprocessor.

The Coprocessor device driver monitors the status of its communication with the Coprocessor and the Coprocessor hardware status registers. Programs such as the Coprocessor Load Utility (CLU), and the CCA and PKCS #11 Support Programs can receive unusual status in the form of a 4-byte return code from the device driver.

There are a very large number of possible 4-byte codes, all of which are of the form X'8xxxxxx'. The most likely codes that may be encountered are described in Table E-1 on page E-2. If you encounter codes of the form X'8340xxxx' or X'8440xxxx', and the code is not in the Table, contact the IBM 4758 Crypto Team for advice via email from the Support page on the IBM 4758 product Web site (<http://www.ibm.com/security/cryptocards>).

b

b

Table E-1. Device Driver Error Codes in the Class X'8xxxxxx'

4-byte Return Code (hex)	Reason	Considerations
8040FFBF	External intrusion	Arises due to optional electrical connection to the Coprocessor. This condition can be reset.
8040FFDA	Dead battery	The batteries have been allowed to run out of sufficient power, or have been removed. The Coprocessor is zeroized and is no longer functional.
b b 8040FFDB	X-ray tamper or dead battery	The Coprocessor is zeroized and is no longer functional.
b 8040FFDF	X-Ray or dead battery	The Coprocessor is zeroized and is no longer functional.
8040FFEB	Temperature tamper	High or low temperature has been exceeded. The Coprocessor is zeroized and is no longer functional.
8040FFF3	Voltage tamper	The Coprocessor is zeroized and is no longer functional.
8040FFF9	Mesh tamper	The Coprocessor is zeroized and is no longer functional.
8040FFFB	Reset bit is on	Either low voltage was detected, the internal operating temperature of the Coprocessor went out of limits, or the host driver sent a reset command. Try removing and reinserting the Coprocessor into the PCI bus.
8040FFFE	Battery warning	Battery power is marginal. The battery changing procedure described in the <i>IBM 4758 Installation Manual</i> should be followed to replace the batteries.
804xxxx (e.g. 80400005)	General communication problem	Except for the prior X'8040xxxx' codes, there are additional conditions that arise in host-Coprocessor communication. Determine that the host system in fact has a Coprocessor. Try removing and reinserting the Coprocessor into the PCI bus. Run the CLU status command (ST). If problem persists, contact IBM 4758 Support via the Web site.
b b 8340xxxx	Miniboot-0 codes	This class of return code arises from the lowest-level of reset testing. If codes in this class occur, contact the Crypto Support Team via the http://www.ibm.com/security/cryptocards .
8340038F	Random-number generation fault	Continuous monitoring of the random-number generator has detected a possible problem. There is a small statistical probability of this event occurring without indicating an actual ongoing problem. The CLU status (ST) command should be run at least twice to determine if the condition can be cleared.
8440xxxx	Miniboot-1 codes	This class of return code arises from the replaceable POST and code-loading code.
844006B2	Invalid signature	The signature on the data sent from the CLU utility to Miniboot could not be validated by Miniboot. Be sure that you are using an appropriate file (for example, CR1xxxxx .CLU versus CE1xxxxx .CLU). If the problem persists, obtain the output of a CLU status report and forward this and a description of what you are trying to accomplish to Customer Support using the IBM 4758 Web site reporting process.

Appendix F. Master-Key Cloning

This appendix includes:

- A procedure that outlines how to clone a master key from one Coprocessor to another Coprocessor using the CNM utility
- Access-control considerations when cloning.

Master-Key Cloning Procedure

The following procedure outlines how to clone a master key from one Coprocessor to another Coprocessor using the CNM utility. Before using this procedure, you should familiarize yourself with the material presented at “How to Clone a Master Key” on page 5-18 and “Understanding and Managing Master Keys” in Chapter 2 of the *CCA Basic Services Reference and Guide* manual.

The master-key cloning procedure that follows makes no assumption about which computer contains the Coprocessors used for:

- Share administration (“SA node”)
- Master-key source (“CSS” Coprocessor Share-Signing node)
- Master-key target (“CSR” Coprocessor Share-Receiving node).

The SA key can reside in the same Coprocessor as either the CSS or the CSR key, or it can reside in a separate Coprocessor node. Any of the Coprocessors can reside together in the same computer if multiple Coprocessors with CCA are available.

The procedure ignores operator actions to:

- Logon and logoff, as these steps will depend on the specific roles in use at your installation
- Switch between Coprocessors when you are using more than one Coprocessor within a computer.

The procedure is broken down into several phases as outlined in Table F-1.

Phase	Node	Task
1	SA	Establish the Share Administration node; create the SA database, generate the SA key, and store its public key and hash into the SA database.
2a	Source	Establish the source node; generate the “CSS” key and add the public key to the SA database; install the SA public-key.
2b	SA	Certify the CSS key and store the certificate into the SA database.
For each target node, repeat the phase 3 procedures.		
3a	Target	Establish the target node; create a CSR database, generate a “CSR” key and add the public key to the CSR database for this node; install the SA public-key.

Table F-1 (Page 2 of 2). Master-Key Cloning Procedure Phase Overview

Phase	Node	Task
3b	SA	Certify the CSR key and store the certificate into the CSR database for the target node.
3c	Source	Obtain shares and current master-key-verification information.
3d	Target	Install shares and confirm new master-key; set the master key.

Before undertaking the procedure, it is recommended that you complete the forms found on the following pages.

Task	Node	Profile	Role	Responsible Individual
Audit Access Controls	SA			
Generate SA Key	SA			
Register SA-Key Hash	SA			
Register SA Key	SA			
Audit Access Controls	CSS			
Generate CSS Key	CSS			
Obtain CSS Master Key	CSS			
Register SA-Key Hash	CSS			
Register SA Key	CSS			
Certify CSS Key	SA			
Audit Access Controls	CSR1			
Generate CSR Key	CSR1			
Register SA-Key Hash	CSR1			
Register SA Key	CSR1			
Certify CSR1 Key	SA			
Obtain Shares	CSS			
Install Shares	CSR1			
Verify CSR New	CSR1			
Set CSR Master-Key	CSR1			
Audit Access Controls	CSR2			
Generate CSR Key	CSR2			
Register SA-Key Hash	CSR2			
Register SA Key	CSR2			
Certify CSR2 Key	SA			
Obtain Shares	CSS			
Install Shares	CSR2			
Verify CSR New	CSR2			
Set CSR Master-Key	CSR2			

Figure F-1. Cloning Responsibilities, Profiles and Roles

NODE INFORMATION	Node	Machine	Selector Number	Coprocessor Serial Number	Data Base Path and Name
	SA Node Control				(sa.db)
	CSS Node Source				(sa.db)
	CSR Node Target 1				(csr1.db)
	CSR Node Target 2				(csr2.db)

SA-KEY HASH

--	--	--	--

NUMBER OF SHARES

Minimum: "m"	Maximum: "n"
-----------------	-----------------

SHARES DISTRIBUTION

Obtained from:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Installed into CSR-1:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Obtained from:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Installed into CSR-2:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure F-2. Cloning-Information Worksheet

Phase 1: Establish the Share Administration Node

Using the Coprocessor designated as the Share Administration (SA) "node," follow the steps in Table F-2. Note that this Coprocessor can also serve as the master-key source or a master-key target node.

<i>Table F-2. Master-Key Cloning Procedure: Establish SA Node</i>		
Phase	Task	✓
1.1	Audit the appropriateness of the access controls.	
1.2	Perform time synchronization and insure that the authorization (CCA5203.FCV) is installed.	
1.3	Confirm (or install) the master key.	
1.4	Using the facilities of your operating system, erase any prior SA database from the SA database media.	
1.5	If not already established, enter the Environment ID (EID): <ul style="list-style-type: none"> • Crypto Node, Set Environment ID. • Enter the EID, <i>Load</i>. 	
1.6	Generate the SA key: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Create Keys, Share Administration Key. • Accept the default SA public-key and private-key labels, and enter the location and name of the SA database ("sa.db"). • <i>Create</i>. • Record the hash value for use later in the procedure. 	

Phase 2: Establish the Source Node

Using the Coprocessor designated as the master-key source node, follow the steps in Table F-3. Note that this Coprocessor can also serve as the SA node.

Table F-3. Master-Key Cloning Procedure: Establish Source (CSS) Node		
Phase	Task	✓
2a.1	Audit the appropriateness of the access controls.	
2a.2	Perform time synchronization and insure that the authorization (CCA5203.FCV) is installed.	
2a.3	Confirm the Coprocessor serial number: <ul style="list-style-type: none"> • Crypto Node, Status. • <i>Adapter.</i> • Note the serial number, <i>Cancel.</i> 	
2a.4	Confirm (or install) the master key.	
2a.5	Obtain the current master-key-verification information: <ul style="list-style-type: none"> • Master Key, Verify, Current. • <i>Save to transport media, Cancel.</i> 	
2a.6	If not already established, enter the Environment ID (EID): <ul style="list-style-type: none"> • Crypto Node, Set Environment ID. • Enter the EID, <i>Load.</i> 	
2a.7	If not already established, set the number of shares values, “m” and “n”: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Set Number of Shares. • Set the maximum and minimum number of required shares, <i>Load.</i> 	
2a.8	Generate the CSS key: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Create Keys, CSS Key. • Enter the CSS key label (for example, “CSS.KEY”). • Confirm the Coprocessor serial number. • Confirm or enter the SA database name and location. • <i>Create.</i> 	
2a.9	Register the SA public-key hash: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Register Share Administration Key, SA-Key Hash. • Enter the SA database file name and location, <i>Next.</i> • Enter the SA public-key label (or accept the default). • Enter the SA-key hash, <i>Register.</i> 	
2a.10	Register the SA public-key: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Register Share Administration, SA Key. • Enter the SA database file name and location, <i>Next.</i> • Enter the SA public-key label (or accept the default), <i>Register.</i> 	

Phase 3: Establish Target Node and Clone Master Key

Using the designated Nodes, establish the target node and clone the master key following the steps in Table F-4. Note that this Coprocessor can also serve as the SA node.

Table F-4 (Page 1 of 3). Master-Key Cloning Procedure: Establish CSR Node, Clone Master Key			
Phase	Node	Task	✓
At the Target Node...			
3a.1	Target	Audit the appropriateness of the access controls.	
3a.2	Target	Perform time synchronization and insure that the authorization (CCA5203.FCV) is installed.	
3a.3	Target	Confirm the Coprocessor serial number: <ul style="list-style-type: none"> • Crypto Node, Status. • <i>Adapter.</i> • Note the serial number, <i>Cancel.</i> 	
3a.4	Target	Ensure the existence of a (temporary) master key.	
3a.5	Target	If not already established, enter the Environment ID (EID): <ul style="list-style-type: none"> • Crypto Node, Set Environment ID. • Enter the EID (for example, "CSR1 NODE" and extend with spaces to 16 entered characters). • <i>Load.</i> 	
3a.6	Target	If not already established, set the number of shares values, "m" and "n": <ul style="list-style-type: none"> • Crypto Node, Share Administration, Set Number of Shares. • Set the maximum and minimum number of required shares. • <i>Load.</i> 	
3a.7	Target	Using the facilities of your operating system, erase the csr.db data file.	
3a.8	Target	Generate the CSR key: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Create Keys, CSR Key. • Enter the CSR key label (for example, "CSR1.KEY"). • Confirm the Coprocessor serial number. • Select the key size. • Provide the CSR database name and location (for example, "CSR1.DB"). • <i>Create.</i> 	
3a.9	Target	Register the SA public-key hash: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Register Share Administration, SA-Key Hash. • Enter the SA database file name and location, <i>Next.</i> • Enter the SA public-key label (or accept the default). • Enter the SA-key hash, <i>Register.</i> 	
3a.10	Target	Register the SA public-key: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Register Share Administration Key, SA Key • Enter the SA database file name and location, <i>Next.</i> • Enter the SA public-key label (or accept the default), <i>Register.</i> 	

Table F-4 (Page 2 of 3). Master-Key Cloning Procedure: Establish CSR Node, Clone Master Key

Phase	Node	Task	√
At the SA Node...			
3b.1	SA	Certify the CSS key (as required): <ul style="list-style-type: none"> • Crypto Node, Share Administration, Certify Keys, CSS Key. • Enter the name and path for the SA database, <i>Next</i>. • Confirm the CSS key label, the Coprocessor serial number, and the SA Environment ID. • <i>Certify.</i> 	
3b.2	SA	Certify the CSR key: <ul style="list-style-type: none"> • Crypto Node, Share Administration, Certify Keys, CSR Key. • Enter the name and path for the SA and CSR databases. • <i>Next.</i> • Confirm the SA key label, CSR key label, and the SA Environment ID. • Enter the CSR serial number. • <i>Certify.</i> 	
At the Source Node...			
3c.1	Source	Obtain at least “m” of “n” shares. Perform the following for each share. Note that logon and logoff may be required to obtain each share. <ul style="list-style-type: none"> • Crypto Node, Share Administration, Get Share. • Select the share. Note that if you are obtaining an additional set(s) of shares, the “Distributed” messages may not be meaningful. • Enter the name and path for the SA and CSR databases. • <i>Next.</i> • Confirm the CSS key label, CSS Coprocessor serial number, and the CSR Coprocessor serial number. • <i>Get Share.</i> Repeat as required.	

Table F-4 (Page 3 of 3). Master-Key Cloning Procedure: Establish CSR Node, Clone Master Key			
Phase	Node	Task	√
At the Target Node...			
3d.1	Target	<p>Install “m” (of “n”) shares. Perform the following for each share and observe the response. The response indicates when enough shares have been installed to form the new master-key. Note that logon and logoff may be required to install each share.</p> <ul style="list-style-type: none"> • Crypto Node, Share Administration, Load Share. • Select the share. • Enter the name and path for the CSR and SA databases. • <i>Next.</i> • Confirm the CSS key label, the CSS Coprocessor serial number, and the CSR Coprocessor serial number. • <i>Load Share.</i> <p>Observe the response. Loading sufficient shares completes the new master key. Repeat as required.</p>	
3d.2	Target	<p>Confirm the new master-key:</p> <ul style="list-style-type: none"> • Master Key, Verify, New. • <i>Compare</i>, select the file, <i>OK</i>, <i>Cancel</i> 	
3d.3	Target	<p>Using the facilities of your operating system, erase the csr.db data file. This is not a security issue but rather to avoid complications should you perform another master-key cloning operation.</p>	
3d.4	Target	<p>As appropriate, set the master key:</p> <ul style="list-style-type: none"> • Master Key, Set • <i>OK</i> 	

Access-Control Considerations when Cloning

There are three classes of roles to consider for cloning operations:

- Roles at the SA, share-administration node
- Roles at the source node, CSS
- Roles at the target node, CSR.

And, you must consider your security policy.

Your security policy must define:

- Who will have the authority to generate a random master-key at the source node.
- Who will have the authority to set the master key, the action which brings a new master key into operation. Note that keys enciphered by the master key will need to be updated to the changed master key; you need to plan for this action and which role will provide this authority.
- Who will have the authority to generate the Retained RSA keys to certify the public keys of the source and target nodes (the SA key), and to generate the Retained keys at the source (CSS) and target (CSR) nodes.

- Who will have the authority to register the SA key and its hash, and if this will be a split responsibility.
- How many individuals must cooperate to clone a master key; of course they must be selected to avoid collusion.

In deciding the “m” and “n” values, you must consider when the cloning will take place, and if there is the need to reconstitute the master key from a fewer number of shares than the total number obtained from the source node (perhaps because of share-corruption or the unavailability of one or more individuals who can obtain or install a share).

Note: The CNM utility places all of the shares from a node into the CSR.DB file. Each share is encrypted under a unique, triple-length DES key which itself is encrypted by the CSR public key of the target node.

Table F-5 provides guidance for selecting the permissions applicable to roles related to cloning.

<i>Table F-5 (Page 1 of 2). CCA Commands Related to Master-Key Cloning</i>			
Code	Command Name	Verb Name	Consideration / Role
X'001A'	Set Master Key	Master_Key_Process	Critical, must be knowledgeable of the contents of the new master-key register and the implications of a master-key change.
X'001D'	Compute Verification Pattern	Many	All
X'0020'	Generate Random Master Key	Master_Key_Process	Not critical except that it fills the new master-key register.
X'0032'	Clear New Master Key	Master_Key_Process	Probably assigned to the role that can set the master key. This can override the collected shares. Probably should be mutually exclusive with the Generate Random Master Key command.
X'0033'	Clear Old Master Key Register	Master_Key_Process	Generally not used
X'008E'	Generate Key	Key_Generate† Random_Number_Generate	All
X'0090'	Re-encipher to Current Master Key	Key_Token_Change	Consider who will update working keys encrypted by the master key.
X'0100'	PKA96 Digital Signature Generate	Digital_Signature_Generate	Certifier of the SA, CSS and CSR keys
X'0101'	PKA96 Digital Signature Verify	Digital_Signature_Verify	All
X'0102'	PKA96 Key Token Change	PKA_Key_Token_Change	Consider who will update working keys encrypted by the master key.
X'0103'	PKA96 PKA Key Generate	PKA_Key_Generate	Required to generate the SA, CSS, and CSR keys
X'0107'	One-Way Hash, SHA-1	One_Way_Hash	All
X'0114'	Change User Profile Authentication Data	Access_Control_Initialization	Allows changing the passphrase in ANY profile; use with discretion.
X'0116'	Read Public Access-Control Information	Access_Control_Maintenance	All
X'011C'	Set EID	Cryptographic_Facility_Control	Needed to setup the CSS and CSR nodes.
X'011D'	Initialize Master Key Cloning	Cryptographic_Facility_Control	Needed to setup the “m-of-n” values at the CSS and CSR nodes.
X'0200'	PKA Register Public-Key Hash	PKA_Public_Key_Hash_Register	Use at the CSS and CSR nodes to ensure the SA key can be recognized; split responsibility with X'0201'.
X'0201'	PKA Public Key Register with Cloning	PKA_Public_Key_Register	Use at the CSS and CSR nodes to ensure the SA key can be recognized; split responsibility with X'0200'.
X'0203'	Delete Retained Key	Retained_Key_Delete	Use to remove obsolete SA, CSS, and CSR keys; be careful about denial of service.

Table F-5 (Page 2 of 2). CCA Commands Related to Master-Key Cloning

1

Code	Command Name	Verb Name	Consideration / Role
X'0204'	PKA Clone Key Generate	PKA_Key_Generate	Needed to generate the CSS and CSR keys.
X'0211' through X'021F'	Clone-info (Share) Obtain	Master_Key_Distribution	Consider a profile and role for each share to enforce split responsibility.
X'0221' through X'022F'	Clone-info (Share) Install	Master_Key_Distribution	Consider a profile and role for each share to enforce split responsibility.
X'0230'	List Retained Key	Retained_Key_List	All

b Appendix G. Threat Considerations for a Digital-Signing Server

b This appendix addresses threats which should be considered when employing the
b IBM 4758 with the CCA Support Program in a Digital-Signing Application. Much of
b the discussion is applicable to other environments in which you may apply the
b Coprocessor.

b An organization placing a CA, RA, OCSP responder, or time-stamping service into
b operation should consider how their installation will address various threats.
b Table G-1 on page G-2 lists potential threats and presents product design and
b implementation solutions to many of these threats. Notes are included describing
b steps that you should consider to further mitigate your exposure to problems.

b For the case of a "digital-signing server," "Digital-Signing Server" on page 6-12
b describes actions you should use in deploying the Coprocessor, policies you should
b consider, application-functionality which should be included, and so forth.

b Plan to re-read the contents of Table G-1 after you have made first-pass decisions
b about your installation.

b Table G-1 (Page 2 of 9). Threat Considerations for a Digital-Signing Server	
b	Threat Discussion
b	Threat Mitigation
b	Physical Modification of the Coprocessor
b	An adversary may physically modify the Coprocessor in order to reveal design or security related information. This modification may be achieved through techniques commonly employed in hardware failure analysis and reverse engineering efforts. The goal is to identify such design details as hardware security mechanisms, access-control mechanisms, authentication systems, data protection systems, memory partitioning, or cryptographic programs. Determination of software design, including initialization data, passwords, or cryptographic keys may also be a goal.
b	The sensitive electronics are all packaged within the tamper-responding package mounted on the Coprocessor. The independent physical penetration analysis performed as part of the FIPS 140 evaluation has confirmed that the Model 2 device is exceptionally resistant to undetected alteration. In the process of altering the sensitive electronics, the Coprocessor factory certification would be destroyed rendering the device useless. Note: You will need to physically confirm that a specific, serial-numbered Coprocessor is in use and audit its status-query response to confirm that it remains an unaltered IBM Coprocessor loaded with appropriate software.
b	Environmental Manipulation of the Coprocessor
b	An adversary may utilize environmental conditions beyond those of the Coprocessor specification to obtain or modify data or program flow for fraudulent Coprocessor use. This modification may include manipulation of power lines, clock rates, or exposure to high and low temperatures and radiation. As an effect, the Coprocessor might get into a situation where instructions are not correctly executed. As a result, security-critical data might get modified or disclosed in contradiction to the security requirements for the Coprocessor.
b	The Coprocessor has sensors to detect environmental stresses which might induce erroneous operation. Abnormal conditions as described in Chapter 2 of the <i>IBM 4758 PCI Cryptographic Coprocessor General Information Manual</i> will cause the unit to zeroize.
b	Substituted Process
b	Requests to, and responses from, the Coprocessor may be directed to an alternative implementation enabling an adversary to influence results. An alternative implementation might be substituted with differing security features. For example, private key generation and the production of digital signatures might be performed in an alternative implementation that would enable exposure of the private key.
b	Notes:
b	1. Auditors need to complete the processes described for them to ensure that the signing key is indeed retained within the appropriate Coprocessor.
b	2. Access to the host system should be supervised so that host-system security measures and proper operation can be relied upon.
b	Threats associated with logical attack on the Coprocessor
b	Insertion of Faults
b	An adversary may determine security critical information through observation of the results of repetitive insertion of selected data. Insertion of selected inputs followed by monitoring the output for changes is a relatively well-known attack method for cryptographic devices. The intent is to determine information based on how the Coprocessor responds to the selected inputs. This threat is distinguished by the deliberate and repetitive choice and manipulation of input data as opposed to random selection or manipulation of the physical characteristics involved in input/output operations.
b	The electronic design of the Coprocessor renders classical approaches to smart card attacks infeasible. Note: Supervision of the host system and controlling access to the system, both logically and physically, are important security steps to be taken by the using organization.

b Table G-1 (Page 3 of 9). Threat Considerations for a Digital-Signing Server		
b	Threat Discussion	Threat Mitigation
b	<p>Forced Reset</p> <p>An adversary may force the Coprocessor into a non-secure state through inappropriate termination of selected operations. Attempts to generate a non-secure state in the Coprocessor may be made through premature termination of transactions or communications between the Coprocessor and the host, by insertion of interrupts, or by inappropriate use of interface functions.</p>	<p>The Coprocessor is designed to always run through its initial power-on sequence in the event of trap and reset conditions. Each application-level request is treated as a separate unit of work and processed from a single defined set of initial conditions.</p>
b	<p>Invalid Input</p> <p>An adversary or authorized user of the Coprocessor may compromise the security features of the Coprocessor through introduction of invalid inputs. Invalid input may take the form of operations which are not formatted correctly, requests for information beyond register limits, or attempts to find and execute undocumented commands. The result of such an attack may be a compromise in the security functions, generation of exploitable errors in operation, or release of protected data.</p>	<p>Transaction requests carry authentication information applied in the caller's domain and validated by the Coprocessor. Each request is processed from a single, known state with predefined conditions. The Coprocessor software validates the characteristics of each request to address misuse scenarios.</p>
b	<p>Data Loading Malfunction</p> <p>An adversary may maliciously generate errors in set-up data to compromise the security functions of the Coprocessor. During the stages of Coprocessor preparation which involve loading the Coprocessor with special keys, identification of roles, and so forth, the data itself may be changed from the intended information or may be corrupted. Either event could be an attempt to penetrate the Coprocessor security functions or to expose the security in an unauthorized manner.</p>	<p>Note: As outlined in auditor procedures, the access-control setup should be verified along with confirming the installed Coprocessor software.</p>
b	<p>Unauthorized Program Loading</p> <p>An adversary may utilize unauthorized programs to penetrate or modify the security functions of the Coprocessor. Unauthorized programs may include the execution of legitimate programs not intended for use during normal operation or the unauthorized loading of programs specifically targeted at penetration or modification of the security functions.</p>	<p>The Coprocessor will only accept digitally signed software after the signature has been validated. An independent evaluation of IBM's software build and signing procedures and the Coprocessor design affirms the trust which can be placed in the identity of loaded software.</p> <p>Note: An auditor should follow procedures to affirm that specified software is in use.</p>
b Threats associated with control of access		
b	<p>Invalid Access</p> <p>A user or an adversary of the Coprocessor may access information or services without having permission as defined in the role profile. Each role has defined privileges which allow access only to selected services of the Coprocessor. Access beyond those specified services could result in exposure of secure information.</p>	<p>An auditor can confirm the permissions granted in each established role and the set of profiles (users) associated with each role. An independent evaluation of the Coprocessor software implementation and testing has reviewed the integrity of the access-control implementation.</p>

Table G-1 (Page 6 of 9). Threat Considerations for a Digital-Signing Server	
Threat Discussion	Threat Mitigation
<p>Forging Data Before It Is Signed</p> <p>An adversary may modify data to be digitally signed by the Coprocessor before the signature is generated within the Coprocessor. This attack may use weaknesses in the implementation that allow an adversary to modify data transmitted for signature to the Coprocessor before the Coprocessor actually calculates the signature.</p>	<p>Requests from user host-application process memory carry an integrity check value which the Coprocessor confirms prior to incorporating the hash in a digital signature.</p> <p>Note: Users must review host-system and host-application program security to ensure that authenticated hash values received into the Coprocessor have not been compromised and are representative of the data to be protected.</p>
<p>Misuse of Signature Function</p> <p>An adversary may misuse the Coprocessor signature creation function to sign data the Coprocessor is not supposed to sign.</p> <p>The adversary may try to submit data to the Coprocessor and get it signed without passing the authorization checks of the Coprocessor it has to perform before generating a digital signature.</p> <p>As an alternative an adversary may try to modify data within the Coprocessor through the use of Coprocessor functions or by trying to influence the Coprocessor such that the data in the Coprocessor gets modified.</p>	<p>An independent review of the Coprocessor software affirms that:</p> <ul style="list-style-type: none"> The digital signature generation service requires an appropriate permission in a role The processing of requests and the integrity of the design to prevent data alteration. <p>Notes:</p> <ol style="list-style-type: none"> The integrity of the Coprocessor and its code must be affirmed by an auditor who reviews a Coprocessor status query. An auditor must confirm that appropriate access-control roles and profiles have been established which exclude unauthorized users from use of the digital signing functionality.
<p>Forging Signature Verification Function</p> <p>An adversary may modify the function for signature verification such that a false signature is accepted as valid. This attack may try to modify the signature verification function or signed data to be verified such that the Coprocessor returns a success message when this false signature is presented to him for verification.</p>	<p>The signature verification function of primary interest here occurs in the Coprocessor's code-loading process (in "Miniboot"). With this product:</p> <ul style="list-style-type: none"> Miniboot code, like the control program and (CCA) application program code, is only accepted into the Coprocessor when the Coprocessor validates the signature on the signed code. The initial Miniboot code loaded in the factory is also subject to digital signature verification. Standardized cryptographic processes are used (SHA-1, RSA, ISO 9796) for the signature. The code building and signing process are the subject of an independent review.
<p>Disclosure of a Private RSA Signature Key</p> <p>An adversary may use a function(s) that discloses a private RSA signature key.</p>	<p>An independent evaluation is expect to confirm that the CCA Support Program does not contain any function to output or reveal the value of a retained private key. Certified evaluations have demonstrated that the control program does not output data retained in Coprocessor persistent storage nor is there any lower-level function to read such storage.</p>

b Table G-1 (Page 8 of 9). Threat Considerations for a Digital-Signing Server		
b	Threat Discussion	Threat Mitigation
b Miscellaneous threats		
b	Linked Attacks	Notes:
b	An adversary may perform successive attacks with the result that the Coprocessor becomes unstable or some aspect of the security functionality is degraded. A following attack may then be successfully executed.	1. Use of the cryptographic system should be limited to authorized situations enforced through the Coprocessor access controls and through use of host-system controls.
b	Monitoring outputs while manipulating inputs in the presence of environmental stress is an example of a linked attack.	2. Host-system controls and organizational policies should restrict the access to the system for monitoring and the submission of arbitrary requests.
b	Repetitive Attack	Note: Use of the cryptographic system should be limited to authorized situations enforced through the Coprocessor access controls and through use of host-system controls. Host-system controls and organizational policies should restrict the access to the system for monitoring and the submission of arbitrary requests.
b	An adversary may utilize repetitive undetected attempts at penetration to expose memory contents or to change security critical elements in the Coprocessor. Repetitive attempts related to some or all of the other threats discussed herein may be used to iteratively develop an effective penetration of the Coprocessor security. If these attacks can, in all cases, remain undetected, there will be no warning of increased vulnerability.	
b	Cloning	Note: Auditors must confirm that the digital-signing key, appropriate code, and access-control regime is resident in the authorized Coprocessor.
b	An adversary may clone part or all of a functional Coprocessor to develop further attacks. The information necessary to successfully clone part or all of a Coprocessor may derive from detailed inspection of the Coprocessor itself or from illicit appropriation of design information.	
b Threats Addressed by the Operating Environment		
b	Coprocessor Modification and Reuse	Notes:
b	An adversary may use a modified Coprocessor to masquerade as an original Coprocessor so that information assets can be fraudulently accessed.	1. An auditor must confirm through examination of a Coprocessor-signed query response that the device is genuine and that the appropriate code is loaded.
b	Removal, modification, and re-insertion of that Coprocessor into a host system could be used to pass such a combination as an original. This might then be used to access or change the private signature keys or other security critical information to be protected.	2. The auditor must also confirm that the digital-signing key is a "retained" key in the Coprocessor.
b	Abuse by Privileged Users	Note: An organization must establish, enforce, and audit policies which limit the access that a single individual has to the cryptographic system. The setup procedure must ensure that a single user does not have the opportunity to bring an inappropriate system into production.
b	A careless, willfully negligent, or hostile administrator or other privileged user may create a compromise of the Coprocessor assets through execution of actions which expose the security functions or the protected data. A privileged user or administrator could directly implement or facilitate attacks based on any of the threats described here.	

b Table G-1 (Page 9 of 9). Threat Considerations for a Digital-Signing Server	
b	Threat Discussion
b	Threat Mitigation
b	Data Modification
b	Data to be signed by the Coprocessor may be modified by an adversary or by faults in the operational environment after it has been approved by the legitimate user but before the data is submitted to the Coprocessor to be signed. Data that has been approved by the legitimate user to be signed may be modified by an adversary, by false or malicious programs or by environmental errors (for example, transmission errors) after the data has been approved by the legitimate user and before the data is transferred to the Coprocessor to be signed.
b	Data Verification
b	Signed data to be verified by the Coprocessor may be modified by an adversary or by faults in the operational environment before it is submitted to the Coprocessor for signature verification such that the response of the Coprocessor does not reflect the validity of the signature. Signed data submitted by a user may be modified within the Coprocessor environment before it is passed to the Coprocessor for verification. This may result in a response from the Coprocessor that does not reflect the actual validity of the digital signature that should be verified.
b	There is also the possibility that the response of the Coprocessor is modified in the Coprocessor environment before it is passed to the user that requested the signature verification.
	Note: Host-system security precautions and organization policies must be defined, enforced, and audited to thwart such attacks.
	The Coprocessor verifies the signature on code and certain code-loading commands. The FIPS 140 evaluation confirmed that this cannot be bypassed.
	The CCA design supports validation of the integrity of requests and responses between the Coprocessor and the top layer of CCA code in the host-system.
	Note: Host-system security measures must address blocking the modification of request input and outputs.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights, or other legally protectable rights, may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

Licensors of this program who wish to have information about it for the purpose of enabling (i) the exchange of information between independently created programs and other programs (including this one), and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department VM9A, MG39/201
8501 IBM Drive
Charlotte, NC 28262-8563
U.S.A.

Such information may be available—subject to appropriate terms and conditions—including, in some cases, the payment of a fee.

IBM may have patents or pending-patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

License

You can obtain the files for the CCA Cryptographic Coprocessor Support Program feature by downloading from the Software Download page on the product Web site at <http://www.ibm.com/security/cryptocards>.

The CCA Cryptographic Coprocessor Support Program must be used in accordance with the IBM System Program License Agreement.

Copying and Distributing Softcopy Files

For online versions of this book, we authorize you to:

- Copy, modify, and print the documentation contained on the media, for use within your enterprise, provided you reproduce the copyright notice, all warning statements, and other required statements on each copy or partial copy.
- Transfer the original unaltered copy of the documentation when you transfer the related IBM product (which may be either machines you own, or programs, if the program's license terms permit a transfer). You must, at the same time, destroy all other copies of the documentation.

You are responsible for payment of any taxes, including personal property taxes, resulting from this authorization.

THERE ARE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

Your failure to comply with the terms above terminates this authorization. Upon termination, you must destroy your machine readable documentation.

Trademarks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	AIX/6000
IBM	IBM net.commerce
IBM Registry	IBM World Registry
Operating System/2	Operating System/390
OS/2	OS/390
RS/6000	

The following terms, denoted by a double asterisk (**) in this publication, are the trademarks of other companies:

Adobe Acrobat
Netscape Navigator

RSA

UNIX

VISA

Windows 2000

Windows NT

Java

SET and
Secure Electronic Transaction

Adobe Systems, Inc.
Netscape
Communications Corp.
RSA Data Security,
Inc.
UNIX Systems
Laboratories, Inc.
VISA International
Service Association
Microsoft Corp.
Microsoft Corp.
Sun Microsystems,
Inc.
Trademarks and
service marks owned
by
SET Secure
Electronic Transaction
LLC

List of Abbreviations and Acronyms

ANSI	american national standards institute	I/O	input/output
AIX	advanced interactive executive (operating system)	IPL	initial program load
API	application programming interface	ISO	international organization for standardization
ASCII	american national standard code for information interchange	KEK	key-encrypting key
C	celsius	LU	logical unit
CA	certification authority	MB	megabyte
CBC	cipher block chain	MAC	message authentication code
CCA	common cryptographic architecture	MD5	message digest 5 (hashing algorithm)
CDMF	commercial data masking facility	MDC	modification detection code
CDSA	cryptographic data security architecture	MHz	megahertz
CLU	coprocessor load utility	ODM	object data manager
CNI	cryptographic node initialization	OEM	original equipment manufacturer
CNM	cryptographic node management	PC	personal computer
CP/Q++	control program/q with 4758 extensions	PCI	peripheral component interconnect
CPU	central processing unit	PDD	physical device driver
CSP	cryptographic service provider	PDF	portable document format
CV	control vector	PIN	personal identification number
DEA	data encryption algorithm	PKA	public-key algorithm
DES	data encryption standard	PKCS	public-key cryptography standard
DMA	direct memory access	POST	power-on self-test
ECB	electronic codebook	PPD	program proprietary data
EPROM	erasable programmable read only memory	RAM	random access memory
FCC	federal communications commission	RNG	random-number generator
FCV	function-control vector	ROM	read-only memory
FIPS	federal information processing standard	RSA	rivest, shamir, and adleman (algorithm)
IBM	international business machines	SA	share administration
ICSF	integrated cryptographic service facility	SAA	systems application architecture
		SCC	secure cryptographic coprocessor
		SHA	secure hashing algorithm
		SKA	secret key authentication

Glossary

This glossary includes some terms and definitions from the *IBM Dictionary of Computing*, New York: McGraw Hill, 1994. This glossary also includes some terms and definitions taken from:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) following the definition.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) following the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) following the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

A

access. In computer security, a specific type of interaction between a subject and an object that results in the flow of information from one to the other.

access control. Ensuring that the resources of a computer system can be accessed only by authorized users and in authorized ways.

access method. A technique for moving data between main storage and input/output devices.

advanced interactive executive (AIX) operating system. IBM's implementation of the UNIX** operating system.

american national standard code for information interchange (ASCII). The standard code, using a coded character set consisting of seven-bit characters (eight bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

american national standards institute (ANSI). An organization consisting of producers, consumers, and

general interest groups that establishes the procedures by which accredited organizations create and maintain voluntary industry standards for the United States. (A)

application program interface (API). A functional interface supplies by the operating system or by a separate program that allows an application program written in a high-level language to use specific data or functions of the operating system or the separate program.

authentication. (1) A process used to verify the integrity of transmitted data, especially a message. (T) (2) In computer security, a process used to verify the user of an information system or protected resource.

authorization. (1) In computer security, the right granted to a user to communicate with or make use of a computer system. (T) (2) The process of granting a user either complete or restricted access to an object, resource, or function.

authorize. To permit or give authority to a user to communicate with or make use of an object, resource, or function.

authorized program facility (APF). A facility that permits identification of programs authorized to use restricted functions.

B

bus. In a processor, a physical facility along which data is transferred.

C

card. (1) An electronic circuit board that is plugged into an expansion slot of a system unit. (2) A plug-in circuit assembly.

CDMF algorithm. An algorithm for data confidentiality applications; it is based on the DES algorithm and possesses 40-bit key strength.

ciphertext. (1) Text that results from the encipherment of plaintext. (2) See also *plaintext*.

cipher block chain (CBC). A mode of operation that cryptographically connects one block of ciphertext to the next plaintext block.

cleartext. (1) Text that has not been altered by a cryptographic process. (2) Synonym for *plaintext*. (3) See also *ciphertext*.

common cryptographic architecture (CCA) API. The application program interface described in the *IBM 4758 CCA Basic Services Reference and Guide*, SC31-8609.

control_vector. (1) In the CCA, a 16-byte string that is exclusive-ORed with a master key or a KEK to create another key that is used to encipher and decipher data or data keys. A control_vector determines the type of key and restrictions on its use. (2) See also *key_token*.

coprocessor. (1) A supplementary processor that performs operations in conjunction with another processor. (2) A microprocessor on an expansion card that extends the address range of the processor in the host system, or adds specialized instructions to handle a particular category of operations; for example, an I/O coprocessor, math coprocessor, or a network coprocessor.

cryptographic coprocessor (IBM 4758). An expansion board that provides to a workstation a comprehensive set of cryptographic functions.

cryptographic key data set (CKDS). A data set that contains the encryption keys used by an installation.

cryptographic node. A node that provides cryptographic services, such as key generation and digital signature support.

cryptography. (1) The transformation of data to conceal its meaning. (2) In computer security, the principles, means, and methods used to so transform data.

D

data encrypting key. (1) A key used to encipher, decipher, or authenticate data. (2) Contrast with *key encrypting key*.

data encryption algorithm (DEA). A 64-bit block cipher that uses a 64-bit key, of which 56 bits are used to control the cryptographic process and eight bits are used to check parity.

data encryption standard (DES). The National Institute of Standards and Technology (NIST) Data Encryption Standard, adopted by the U.S. government as Federal Information Processing Standard (FIPS) Publication 46 which allows only hardware implementations of the data encryption algorithm.

decipher. (1) To convert enciphered data into clear data. (2) Contrast with *encipher*.

direct memory access (DMA). The transfer of data between memory and input/output units without processor intervention.

driver. A program that contains the code needed to attach and use a device.

E

electronic codebook (ECB). A mode of operation used with block-cipher cryptographic algorithms in which plaintext or ciphertext is placed in the input to the algorithm and the result is contained in the output of the algorithm.

encipher. (1) To scramble data or to convert data to a secret code that masks the meaning of the data. (2) Contrast with *decipher*.

enciphered data. (1) Data whose meaning is concealed from unauthorized users or observers. (2) See also *ciphertext*.

expansion board. Synonym for *expansion card*.

expansion card. (1) A circuit board that a user can install in an expansion slot to add memory or special features to a computer. (2) Synonym for *card*.

expansion slot. One of several receptacles in a PC or RS/6000 machine into which a user can install an expansion card.

exporter key. (1) In the CCA, a type of DES KEK that can encipher a key at a sending node. (2) Contrast with *importer key*.

F

feature. A part of an IBM product that can be ordered separately.

federal information processing standard (FIPS). A standard that is published by the US National Institute of Science and Technology.

first in first out (FIFO). A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time. (A)

flash EPROM. A specialized version of erasable programmable read only memory (EPROM) commonly used to store code in small computers.

function-control vector. A signed value provided by IBM to enable the CCA application in the IBM 4758 PCI Cryptographic Coprocessor to yield a level of cryptographic service consistent with applicable export-and-import regulations.

H

host computer. In regard to the CCA Cryptographic Coprocessor Support Program, the workstation into which the IBM 4758 PCI Cryptographic Coprocessor is installed.

I

importer key. (1) In CCA products, a type of DES KEK that can decipher a key at a receiving node. (2) Contrast with *exporter key*.

initial program load (IPL). (1) The initialization procedure that causes an operating system to commence operation. (2) The process by which a configuration image is loaded into storage. (3) The process of loading system programs and preparing a system to run jobs.

inline code. In a program, instructions that are executed sequentially, without branching to routines, subroutines, or other programs.

integrated cryptographic service facility (ICSF). An IBM-licensed program that supports the cryptographic hardware feature in the MVS environment for the high-end System/390* processor.

interface. (1) A boundary shared by two functional units, as defined by functional characteristics, signal characteristics, or other characteristics as appropriate. The concept includes specification of the connection between two devices having different functions. (T) (2) Hardware, software, or both, that links systems, programs, and devices.

international organization for standardization (ISO). An organization of national standards bodies established to promote the development of standards to facilitate the international exchange of goods and services, and to foster cooperation in intellectual, scientific, technological, and economic activity.

J

K

key. In computer security, a sequence of symbols used with an algorithm to encipher or decipher data.

key encrypting key (KEK). (1) A key used to cipher and decipher other keys. (2) Contrast with *data encrypting key*.

key_label. In CCA products, an indirect identifier for a key_token record in key storage.

key storage. In CCA products, a data file that contains cryptographic keys.

key_token. In CCA products, a data structure that can contain a cryptographic key, its control_vector, and other information related to the key.

L

M

master key. In the 4758's CCA implementation, the key used to encrypt keys to process other keys or data at the node.

megabyte (MB). 1 048 576 bytes.

message authentication code (MAC). In computer security, (1) a number or value derived by processing data with an authentication algorithm, (2) the cryptographic result of block-cipher operations on text or data using the cipher block chain (CBC) mode of operation.

multi-user environment. A computer system that supports terminals and keyboards for more than one user at the same time.

N

national institute of science and technology (NIST). Current name for the US National Bureau of Standards.

node. (1) In a network, a point at which one or more functional units connects channels or data circuits. (I) (2) The endpoint of a link or a junction common to two or more links in a network. Nodes can be processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities.

O

operating system/2 (OS/2). An IBM operating system for personal computers.

P

passphrase. In computer security, a string of characters known to the computer system and to a user; the user must specify it to gain full or limited access to the system and the data stored therein.

plaintext. (1) Data that has not been altered by a cryptographic process. (2) Synonym for *cleartext*. (3) See also *ciphertext*.

power on self test (POST). A series of diagnostic tests that runs automatically when device power is turned on.

private key. (1) In computer security, a key that is known only to the owner and used with a public key algorithm to decipher data. Data is enciphered using the related public key. (2) Contrast with *public key*. (3) See also *public key algorithm*.

procedure call. In programming languages, a language construct for invoking execution of a procedure. (1) A procedure call usually includes an entry name and the applicable parameters.

profile. Data that describes the significant characteristics of a user, a group of users, or one-or-more computer resources.

programmed cryptographic facility (PCF). An IBM-licensed program that provides facilities for enciphering and deciphering data, and for creating, maintaining, and managing cryptographic keys.

public key. (1) In computer security, a key that is widely known and used with a public key algorithm to encipher data. The enciphered data can be deciphered only with the related private key. (2) Contrast with *private key*. (3) See also *public key algorithm*.

public key algorithm (PKA). (1) In computer security, an asymmetric cryptographic process that uses a public key to encipher data and a related private key to decipher data. (2) Contrast with *data encryption algorithm* and *data encryption standard algorithm*. (3) See also *RSA algorithm*.

R

random access memory (RAM). A storage device into which data is entered and from which data is retrieved in a non-sequential manner.

read only memory (ROM). Memory in which stored data cannot be modified routinely.

reduced instruction set computer (RISC). A computer that processes data quickly by using only a small, simplified instruction set.

RSA algorithm. A public key encryption algorithm developed by R. Rivest, A. Shamir, and L. Adleman.

S

secret key authentication (SKA) certificate. The SKA certificate contains enciphered values that could allow IBM to re-initialize a Coprocessor after its tamper-sensors have been triggered. Without a copy of the certificate, there is no way to recover the Coprocessor.

security. The protection of data, system operations, and devices from accidental or intentional ruin, damage, or exposure.

session level encryption (SLE). A Systems Network Architecture (SNA) protocol that provides a method for establishing a session with a key unique to that session. This protocol establishes a cryptographic key, and the rules for deciphering and enciphering information in a session.

system administrator. The person at a computer installation who designs, controls, and manages the use of the computer system.

systems network architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. **Note:** The layered structure of SNA allows the ultimate origins and destinations of information, that is, the end users, to be independent of and unaffected by the specific SNA network services and facilities used for information exchange.

T

throughput. (1) A measure of the amount of work performed by a computer system over a given period of time; for example, number of jobs-per-day. (A) (1) (2) A measure of the amount of information transmitted over a network in a given period of time; for example, a network's data-transfer-rate is usually measured in bits-per-second.

token. (1) A string of characters treated as a single entity. (2) A particular message or bit pattern that signifies permission to transmit. (3) See also *key_token*.

U

utility program. A computer program in general support of computer processes. (T)

V

verb. A function possessing an `entry_point_name` and a fixed-length parameter list. The procedure call for a verb uses the syntax standard to programming languages.

W

Windows (NT). A Microsoft operating system for personal computers.

workstation. A terminal or microcomputer, usually one that is connected to a mainframe or a network, from which a user can perform applications.

Numerics

4758. IBM 4758 PCI Cryptographic Coprocessor.

Index

A

- access-control system
 - considerations when cloning F-9
 - examples 6-11
 - initial state 5-10
 - initialization 5-15
 - list A-1
 - locking 6-2
 - overview 5-10
 - role 5-11
 - security concepts 6-1
 - user profile 5-13
- adjusting the windows NT system time 3-8
- AIX
 - configuration utilities 3-3
 - CSUFADMIN utility 3-3
 - CSUFAPPL utility 3-3
 - CSUFKEYS utility 3-3
 - file permissions 3-4
 - file permissions, default 3-3
 - key-storage locations, default 3-3
 - ODM 3-3
 - ODMGET command 3-4
- AIX utility, odmget 3-4
- application programs
 - compile 7-2
 - link to CCA 7-2
- auditor 4-4
- auto-set, master key 5-17

B

- batteries, coprocessor
 - removal 2-3
 - status 5-9
- battery 2-2

C

- C programming language
 - sample routine 7-3
 - verb calls 7-1
- CCA cryptographic Coprocessor support program
 - See support program
- CCA node initialization utility
 - See NIU (node initialization utility)
- CCA node management utility
 - See NMU (node management utility)
- choosing among Coprocessors 5-7
- clock-calendars, synchronization 5-8
- CLU (coprocessor load utility)
 - commands 4-9

- CLU (coprocessor load utility) (*continued*)
 - overview 4-1
 - return codes 4-11
 - software validation 4-7
 - syntax 4-8
- CNI list 5-2
- code identifiers 6-1
- code levels 6-1
- code-signing node 6-21
- commands, access control
 - See access-control system
- compile, application programs 7-2
- components, support program 3-1
- configuration utilities, AIX 3-3
- configure
 - NMU 5-8
 - permissions, AIX 3-3
- coprocessor
 - installation 2-2
 - load, software 4-1
 - memory segments 4-6
 - polling information 5-9
 - status, batteries 5-9
- Coprocessor load utility
 - See CLU (coprocessor load utility)
- Coprocessor support program
 - See support program
- create
 - KEK 5-23
 - key label 5-23
 - key storage 5-21
 - master key 5-17
 - role 5-11
 - user profile 5-13
- cryptographic key management 5-15
- cryptographic keys 6-4
- csufadmin utility 3-3
- csufappl utility 3-3
- csufkeys utility 3-3

D

- DEFAULT role
 - description 5-10
 - initial use 5-10, B-1
- defaults
 - key-storage locations, AIX 3-3
 - NMU 5-8
 - permissions, AIX 3-3
- define
 - role 5-11
 - user profile 5-13

- delete
 - role 5-12
 - user profile 5-14
- description
 - DEFAULT role 5-10
 - KEKs 5-15
 - master key 5-15
- digital-signing server 6-12
- download, support program 2-3

E

- edit
 - role 5-12
 - user profile 5-14
- environment ID, EID 6-3
- establish owner command 4-7

F

- file permissions, AIX 3-4
- function-control vector 6-3
 - load 5-8

H

- host install, support program
 - See install host software
- host uninstall, support program
 - See uninstall host software

I

- initial state, access-control system 5-10
- initial use, DEFAULT role 5-10, B-1
- initialization
 - access-control system 5-15
 - key storage 5-21
- initialization of the CCA node 5-7
- install host software
 - AIX 3-2
 - NT 3-8
- installation, support program
 - checklist 1-2
 - into Coprocessor 4-1
 - onto host computer 3-1
 - overview 1-1

K

- KEKs
 - create 5-23
 - description 5-15
 - primary 5-15
 - storage 5-23
- key label, create 5-23

- key management, cryptographic 5-15
- key storage
 - create 5-21
 - delete keys 5-22
 - initialization 5-21
 - key label, create 5-23
 - locations, AIX 3-3
 - management 5-21
 - reencrypt 5-22
- key-encrypting keys
 - See KEKs
- key-storage names, verifying in AIX 3-4

L

- link to CCA, application programs 7-2
- list, access-control commands A-1
- load command 4-7
- load Coprocessor software
 - commands 4-9
 - establish owner command 4-7
 - load command 4-7
 - owner identifier 4-7
 - reload command 4-7
 - surrender owner command 4-7
- logon-attempt-failure count, reset 5-15

M

- machine readable log 4-8
- machine-readable log C-1
- make-file 7-2
- management
 - cryptographic key 5-15
 - key storage 5-21
 - master key 5-16
- master key
 - auto-set 5-17
 - create 5-17
 - description 5-15
 - management 5-16
 - new, set 5-17
 - registers 5-16
 - verification 5-16
- master key cloning 6-8
- master-key administration 5-16
- master-key cloning F-1
- memory segments, Coprocessor 4-6
- migrating from Windows NT to 2000 3-11
- migration, version 1 to 2 D-1

N

- NIU (node initialization utility)
 - overview 5-2
 - using, node setup 5-24

- NMU (node management utility)
 - configure 5-8
 - defaults 5-8
 - overview 5-2
- node
 - setup, production-environment 5-4
 - setup, test 5-3

O

- object data manager (ODM) 3-3
- ODM (object data manager) 3-3
- odmget AIX utility 3-4
- ODMGET command 3-4
- ordering
 - batteries 2-2
 - overview 2-1
 - placing orders 2-2
- overview
 - access-control system 5-10
 - CLU 4-1
 - CNI 5-2
 - CNM 5-2
 - installation, support program 1-1
- owner identifier 4-7

P

- PCI Cryptographic Coprocessor
 - See coprocessor
- performance, enhancing 7-8
- permissions, AIX 3-3
- permit, access-control commands 5-11
- PIN data 6-10
- polling information, Coprocessor 5-9
- pre-XOR technique 6-6, 6-8
- primary KEKs
 - See KEKs
- product
- production-environment, node setup 5-4
- profile
 - See user profile
- programs
 - See application programs

R

- reencipher stored keys 5-22
- registers, master key 5-16
- reload command 4-7
- remove host software
 - See uninstall host software
- replicated key 6-8
- reset logon-attempt-failure count 5-15
- restrict, access-control commands 5-11

- return codes, CLU 4-11
- role
 - create 5-11
 - define 5-11
 - delete 5-12
 - edit 5-12
- roles and profiles 6-3

S

- sample routine, C programming language
 - make-file 7-2
 - source code 7-2
 - syntax 7-2
- secret key authentication (SKA) certificate
 - See SKA (secret key authentication) certificate
- secured code-signing node 6-21
- security advice 6-1
- security-relevant data item (SRDI) 4-7
- set new master-key 5-17
- setup
 - production-environment node 5-4
 - test node 5-3
- SKA (secret key authentication) certificate
- software load, Coprocessor
 - See load Coprocessor software
- software validation, CLU 4-7
- SRDI, security-relevant data item 4-7
- status data 6-10
- status, Coprocessor batteries 5-9
- storage, KEKs 5-23
- stored keys, reencipher 5-22
- summary of changes xi
 - refid=order,Coprocessors in pSeries servers 2-2
 - refid=order,Coprocessors in Windows systems 2-1
- support program
 - components 3-1
 - configuration utilities, AIX 3-3
 - Coprocessor load 4-1
 - download 2-3
 - host install 3-1
 - host remove 3-1
 - overview, installation 1-1
- surrender owner command 4-7
- synchronization, clock-calendars 5-8
- syntax
 - CLU 4-8
 - verb calls, C programming language 7-1

T

- test setup, node 5-3
- threat considerations, digital-signing server G-1
- throughput, enhancing 7-8
- TZ, setting Windows time zone 3-8

U

- uninstall host software
 - AIX 3-6
 - NT/2000 3-10
- updating from a prior release in AIX 3-5
- usage security observations 6-1
- user profile
 - create 5-13
 - define 5-13
 - delete 5-14
 - edit 5-14
 - reset logon-attempt-failure count 5-15
- utilities
 - CLU 4-1
 - CNI 5-2
 - CNM 5-2
 - csufadmin 3-3
 - CSUFAPPL 3-3
 - CSUFKEYS 3-3
 - NIU 5-24
 - odmget 3-4

V

- validation, Coprocessor software 4-7
- vector, function-control
 - See function-control vector
- verb calls, C programming language 7-1
- verification, master key 5-16
- verifying key-storage names with AIX 3-4

W

- Windows install and remove 3-7

Z

- zeroization of the CCA node 5-7

IBM

PDF File