IBM Cryptographic Products

CCA Support Program Installation Manual

for IBM 4758 Models 1 & 13 with Release 1.32

Note!

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

1.32 Edition (April 2000)

This edition applies to:

- IBM 4758 Models 001 and 013
- Release 1.3.2.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for IBM AIX.*
- Release 1.32 of the licensed CCA Cryptographic Coprocessor Support Program feature for IBM Operating System/2 Warp.*
- Release 1.32 of the licensed CCA Cryptographic Coprocessor Support Program feature for Microsoft Windows NT.**

Each feature is designed to provide software support for an IBM 4758 PCI Cryptographic Coprocessor, Model 001 or 013, installed into a computer running the concomitant 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 website for an updated version of this publication.

IBM does not stock publications at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office that serves your location. This and other publications related to the IBM 4758 Coprocessor can be obtained in PDF format from the Library page that you can locate at http://www.ibm.com/security/cryptocards.

Readers' comments can be communicated to IBM by using the question and suggestion form on the product website at *http://.www.ibm.com/security/cryptocards*, or by sending a letter to:

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

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

Contents

Figures

About This Publication

This installation and operation guide describes the CCA Cryptographic Coprocessor Support Program feature Release 1.32 for the IBM 4758 PCI Cryptographic Coprocessor Models 001 and 013. This feature includes device drivers, utilities, and the IBM Common Cryptographic Architecture (CCA) application.

The feature is designed to be used with the AIX, OS/2, and Windows NT operating systems.

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
- Use the utilities supplied with the support program to:
  - Load the Coprocessor function-control vector
  - Initialize the Coprocessor
  - 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- 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 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, “Ordering and Obtaining Coprocessor Software” describes how to order and obtain 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, “Device Driver Error Codes” provides error code information that can be observed when operating the CLU utility.

Appendix E, “Notices” includes product and publication notices.

A list of abbreviations, a glossary, and an index complete the manual.

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 website at *http://.www.ibm.com/security/cryptocards*. From the website, 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*.

Other CCA Implementation Publications

For information about the Integrated Cryptographic Service Facility (ICSF) for large servers, refer to:

- *OS/390 Integrated Cryptographic Service Facility Overview*, GC23-3972.

Summary of Changes

Release 1.32 IBM 4758 Models 001 and 013

This edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual* contains information about Release 1.32 (1.3.2.0) and additional information supporting the new IBM 4758 Model 13 PCI Cryptographic Coprocessor that is certified by NIST for compliance with the FIPS PUBS 140-1 standard at level 3.

Release 1.32 incorporates these changes from the prior release, 1.31:

1. A new device driver fix for AIX
2. Removal of extra Coprocessor resets that slowed operation of CLU
3. Provides missing files for use of the CLU VA command
4. Fixes and enhancements to the CCA implementation:
   - Check for the CLONE permission bit in RSA key when cloning master key, and properly set the CLONE bit when generating the RSA key pair.
   - Add a missing access control check to DES Key_Generate verb for SINGLE-R.
   - Separate access control checking for CLEAR and CLONE options in the PKA_Key_Generate verb.
   - Change to the CHGEXPDT mode of the Access_Control_Initialization verb option with PROTECTD option to yield an error return of 8/85 and not 0/0.
   - Change to the Access_Control_Initialization verb to return 8/68 and not 8/72 for verb_data_2 length of 0.
   - Change to the Access_Control_Initialization verb so that changing the profile expiration data works properly.
   - Changing a passphrase changes the current user's passphrase and not that of the specified user.
   - Change to the Master_Key_Distribution verb to return 8/1030 when using the Master_Key_Distribution verb after the master key is changed.
   - Change to the Access_Control_Initialization verb so that a very large role count will not cause a very long delay in receiving the verb response.

- Change to the Master_Key_Process verb so that the M-of-N setting is not altered by this verb.
- Change to the Key_Generate verb so that it does not alter a DATAM control vector in an external key token.
- Change the Logon_Control verb so that a forced logoff works correctly.

Other corrections and improvements may be marked with the change bar as at the left. See especially:

- "CCA Support Program and AIX File Permissions" on page 3-4
- The ODMGET command in "How to Configure the Support Program" on page 3-3
- Chapter 6, "Observations on Secure Operations"
- "Additional Application Program Termination Procedure in AIX" on page 7-2.

Release 1.31

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

- Release 1.3.1.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for AIX.
- Release 1.31 of the licensed CCA Cryptographic Coprocessor Support Program feature for OS/2.
- Release 1.31 of the licensed CCA Cryptographic Coprocessor Support Program feature for NT.

Release 1.31 provides maintenance fixes for the previous release 1.30. The major operational change relates to the Coprocessor Load Utility (CLU) described in Chapter 4, "Loading Software into the Coprocessor." The commands that you use to load or reload the Coprocessor software change with this release. Be sure to review the chapter. Always be careful to observe the code loading procedures supplied with the actual software release.

Chapter 4, "Loading Software into the Coprocessor" is substantially modified as the Coprocessor Load Utility user interface is extensively changed. This chapter should be reviewed for changes to your Coprocessor setup procedures.

Release 1.30

The third edition of the *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual*, SC31-8610, contains product information that is current with:

- Release 1.3.0.0 of the licensed CCA Cryptographic Coprocessor Support Program feature for AIX.
- Release 1.30 of the licensed CCA Cryptographic Coprocessor Support Program feature for OS/2.
- Release 1.30 of the licensed CCA Cryptographic Coprocessor Support Program feature for NT.

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; it is used to enable the CCA application within the Coprocessor to yield a level of cryptographic service consistent with applicable import and export regulations.

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

1. **Order and Obtain the Software:** Chapter 2, “Ordering and Obtaining Coprocessor Software” describes how to order the software from IBM, how to download it through the Internet, and how to unpack the downloaded files.
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 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 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 and which function control vector (FCV) are appropriate to your setup: Platform: AIX=4374( ) Windows NT=4376( ) OS/2=4372( ) FCV: 5200 ( ) 5201 ( ) 5202 ( )	"How to Choose Product Features" 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.) IBM Customer Number: _____ (obtain when ordering)	"How to Order and Obtain the IBM 4758 Hardware" on page 2-2	
3	Submit a license key request using the form from the Order page of the product website, *http://www.ibm.com/security/cryptocards*.	"How to Obtain the CCA Support Program License Keys" on page 2-2	
4	Receive license keys attached to e-mail from IBM: Platform License Key: _____ FCV License Key: _____	"How to Obtain the CCA Support Program License Keys" on page 2-2	
5	Receive Coprocessor hardware		
6	Install Coprocessor hardware	"How to Install Your IBM 4758 Hardware" on page 2-3	
7	Download the support program for your operating system and your FCV. Run the downloaded files; you are prompted to supply your IBM Customer Number and license key data.	"How to Download and Decipher the Software and FCV" on page 2-3	
8	Install the support program onto the Coprocessor host computer.	Chapter 3, "Installing the Support Program"	
9	Load Coprocessor software.	Chapter 4, "Loading Software into the Coprocessor"	
10	Setup 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	
11	Run test programs that utilize the CCA libraries.		

Chapter 2. Ordering and Obtaining Coprocessor Software

The CCA Cryptographic Coprocessor Support Program feature is available for download through the Internet from the Order page reached from *http://.www.ibm.com/security/cryptocards*. The software is enciphered; to use it, you must place an order and then obtain the license keys necessary to decipher the support program files. This chapter describes how to:

- Choose the product features you need.
- Order the hardware and software.
- Download the software.

How to Choose Product Features

Before ordering the CCA Cryptographic Coprocessor Support Program, review the following lists to determine the feature codes you need to order:

Note: The support program requires an installed Coprocessor, and the installed Coprocessor cannot function without support software.

1. From the table below, choose the Coprocessor and the (optional) battery kit:

Description	Machine Type	Model No.	Feature Code
PCI Cryptographic Coprocessor FIPS 140-1, level 4	4758	001	
PCI Cryptographic Coprocessor FIPS 140-1, level 3	4758	013	
Replacement Battery Kit			1008

Note:

The battery kit contains two batteries and a temporary-battery tray. The shelf-life of the batteries is nearly the same as the useful life of batteries mounted in an IBM 4758 that is continuously powered on. As a general guideline, consider changing the batteries every three years as a planned event. The actual life of the batteries is anticipated to be in excess of five years. When you do change batteries, be sure that these are fresh and have not been in inventory for a long period.

2. From the table below, choose the support program feature codes for the IBM 4758 Models 1 and 13 that support your operating system(s). The same software is used for a Model 13 as for the Model 1.

Support Program	Feature Code
for AIX **Note:** You must complete support arrangements for specific RS/6000 machines by ordering a no-charge RPQ; see the 4758's *Tested Systems* web page. The RPQ ensures that IBM support services are prepared to support your use of the IBM 4758.	4374
for OS/2	4372
for NT	4376

3. As permitted by import and export regulations, choose *one* function-control vector from the table below. The function-control vector controls the strength of the cryptographic services offered by the CCA software; this strength varies according to specific government regulations.

   An IBM export-regulation coordinator can help you determine the limitations that apply to you, and help you apply for an exception to the standard practice if you need one.

   Note: Under USA Export regulations as of 14 April, 2000, IBM can generally export FCV 5200 to all customers. Import regulations (for example, in France) may restrict the permissible FCV.

Function-Control Vector	Feature Code
DES and CDMF[1], 1024-bit RSA symmetric-key management, and SET[2]	5200
DES and CDMF, 512-bit RSA symmetric-key management, and SET	5201
CDMF, 512-bit RSA symmetric-key management, and SET	5202

How to Order and Obtain the IBM 4758 Hardware

To order the Coprocessor hardware, contact your local IBM representative or your IBM Business Partner, and order the features you have chosen. Customers in the U.S.A. can also contact IBM Direct at 1-800-IBM-CALL.

Ask for the IBM Customer Number that applies to your order. Enter this number in Table 1-1 on page 1-2.

How to Obtain the CCA Support Program License Keys

Request the license keys for your software by submitting the electronic form from the Order page of the product website at *http://.www.ibm.com/security/cryptocards*. IBM will verify your hardware order and will—within several days—deliver your license keys by e-mail.

Each license key is related to the customer number you received from your IBM Representative or IBM Business Partner; you will need these keys and your customer number to decipher the files you download from the product website. You need a license key for *every* operating system feature you order and an additional license key for your function-control vector (FCV).

Note: If you already have a License Key for a given feature code, perhaps because you are obtaining a software update, you do not need to submit a new request.

1 Commercial Data Masking Facility (CDMF) is a DES-based data confidentiality algorithm with a key strength equivalent to 40 bits.

2 SET and Secure Electronic Transaction are trademarks and service marks owned by SET Secure Electronic Transaction LLC.

How to Install Your IBM 4758 Hardware

The IBM 4758 is installed in a manner similar to other PCI boards. You should follow the process described in the *IBM 4758 PCI Cryptographic Coprocessor Installation Manual.*

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.* The Coprocessor is certified at the factory. If it ever detects tampering, or if battery power and system power are removed, the factory certification will be zeroized and the Coprocessor will be rendered non-functional. There is no recovery from this situation.

On AIX systems, it is essential that you first install the hardware. To install correctly, the AIX software installation process must locate a Coprocessor during device driver installation.

How to Download and Decipher the Software and FCV

Once you possess your license keys, you are ready to download through the Internet the support program software and function control vector (FCV).

Tip: To be sure you receive the latest version of the support program, wait to download until you have received and installed your Coprocessor. At that time you should also check the website for any available fix packs.

Note to AIX and NT Users: The unpacking process requires supervisory authority. On AIX, it must be performed by a user with root-level access. On NT, it must be performed by a user with the administrator privilege.

1. Download the operating system features and the function-control vector you ordered from the Order page of the product website at *http://.www.ibm.com/security/cryptocards*. Note that the software is enciphered, and cannot yet be used.

   Note to AIX Users: Verify that the permissions for the downloaded files are configured so that you can execute the files; use the **chmod** command to change the existing permissions.

2. Run the downloaded files. For each file:

   a. You are prompted to enter your customer number and the license key for that file.

   b. The program combines these numbers into a decryption key using a one-way cryptographic algorithm.

   c. The program deciphers the file using the decryption key:

      - On AIX systems, the program also unpacks the support program file, resulting in two install image files: **csuf.bff** and **devices.pci.14109f00.bff** The FCV file unpacks to one of **CCA5200.FCV**, **CCA5201.FCV**, or **CCA5202.FCV**.
      - On OS/2 and NT systems, the resulting file is an executable.

3. If you plan to use the support program on multiple host computers, you can copy the install images or the executable file 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 and deciphering the software as described in Chapter 2, "Ordering and Obtaining Coprocessor Software," follow the procedures in this chapter to install the CCA Cryptographic Coprocessor Support Program onto the Coprocessor host computer. (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.")

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 uninstall the software.

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 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."

When you use the CCA support program you will normally accumulate keys in the CCA key-storage files. Also, through use of the Cryptographic Node Management (CNM) and Cryptographic Node Initialization (CNI) utilities, you may accumulate edited role and profile information unique to your installation. Prior to removing or uninstalling the CCA Support Program, give consideration to saving information that remains important to your applications.

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 uninstall the CCA Support Program for AIX. Following installation of release 1.3.0.0, you update your installation with update 1.3.2.0. You can also update a 1.3.1.0 installation with the 1.3.2.0 update.

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 RS/6000 computer 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*, SC31-8623.

Notice

You must complete support arrangements for specific RS/6000 machines by ordering a no-charge RPQ; see the 4758's *Tested Systems* web page. The RPQ ensures that IBM support services are prepared to support your use of the IBM 4758.

Software:

AIX Version 4.1.5, 4.2, or 4.3 (32-bit mode only)

Java** runtime environment 1.1.2 through 1.1.6, available from *http://www.ibm.com/java/jdk/download*. This is required to use the Cryptographic Node Management Utility.

Disk Space:

4 MB in the */usr* file system.

How to Install the Support Program Base Release 1.3.0.0

First, install the support program base release 1.3.0.0 as described here. Then apply the update 1.3.2.0 as described at “How to Install the Support Program Update 1.3.2.0” on page 3-3.

1. Log on as **root**.
2. Enter the command **smitty cfgmgr**; 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 and Decipher the Software and FCV” on page 2-3; the software is installed.
4. To confirm successful installation, enter the command **lsdev -C -l crypt0**; the system message should reflect status `Available`.

5. Read or print */usr/lpp/csuf/README*; this file contains the latest information about the support program product.
6. Use the configuration utilities to configure the software; those utilities are described in the next section, “How to Configure the Support Program.”

How to Install the Support Program Update 1.3.2.0

This section describes how to apply an update to the base release of the Support Program.

1. If not already installed, first install the base release 1.3.0.0 software. You can also follow this procedure if you have previously updated to release 1.3.1.0.
2. Download the the software update. This file is not encrypted and you do *not* need to use a license key as described in “How to Download and Decipher the Software and FCV” on page 2-3.
3. Log on as **root**.
4. Enter the command **smitty update_all** or **smit update_all**. You are prompted to enter the location of the software update file. Enter the location and press Enter.
5. Change the menu selections so that they are appropriate for your company's installation policy (for example: commit, don't commit, and so forth).
6. Enter the command **lslpp -l csuf.com devices.pci.14109f00.rte** and verify that the version 1.3.2.0 software is installed.
7. Either reboot the machine, or
   a. Enter the command **rmdev -l crypt0**. The system should respond with *crypt0 Defined*.
   b. Enter the command **mkdev -l crypt0**. The system should respond with *crypt0 Available*.
8. Read or print the **/usr/lpp/csuf/README** file. This file contains the latest information about the support program product.
9. Use the configuration utilities to configure the software; those utilities are described in the section “How to Configure the Support Program.”.
10. Update the IBM 4758 internal software according to the instructions in Chapter 4, “Loading Software into the Coprocessor,” and in the README file.

How to Configure the Support Program

The Support Program provides three utility commands you can use to configure the software. For more detail, refer to the AIX **man** page for each utility. You can also use the AIX **odmget** command.

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.

odmget Verifying key storage names with the odmget 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

Key Storage File Names: When you initialize key storage using the Cryptographic Node Management Utility (see “How to Create or Initialize Key Storage” on page 5-21) or with the Key_Storage_Initialize verb, ensure that you specify the file names and directories as registered for these files in the ODM. You can use **odmget** to determine the file names. The ODM entries may be changed using the **csufkeys** command.

Key Record List Output: The CCA Key_Record_List verb and PKA_Key_Record_List verb produce files in the /usr/lpp/csuf/csufkeys/deslist and /usr/lpp/csuf/csufkeys/pkalist directories. 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 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.

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 execute 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----).

Where to Locate RS/6000 Coprocessor Hardware Errors

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

```
errpt -a -N crypt0,libscc.a | more.
```

How to Remove the Support Program

Important: 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. Also remember to back up or save any edited role and profile information.

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.
3. Enter the command **smitty install_remove**; you are prompted to enter the product names.
4. Enter the product names **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.

How to Install and Remove the Support Program for OS/2

This section includes a description of the support program system requirements and procedures necessary to install and uninstall the software.

OS/2 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*, SC31-8623.

Software:

OS/2 WARP Version 4.0, WARP 4.0 Server, and WARP 4.0 Server SMP are supported.

Java runtime environment 1.1.1 through 1.1.7, available from *http://www.ibm.com/java/jdk/download*. This is required to use the Cryptographic Node Management Utility.

Netscape Navigator** for OS/2 Warp Version 2.02 or higher, available from *http://www.internet.ibm.com/browsers/netscape/warp*. This is required to use the install program.

Disk Space:

6.7 MB.

How to Install the Support Program

To install the support program:

1. If a prior release is installed, you must remove the earlier release. Be sure to backup or save the key-storage files, roles, profiles, and any other data or code you are storing in the same directories.
2. From the OS/2 window, go to the directory containing the Coprocessor software package you obtained as described in “How to Download and Decipher the Software and FCV” on page 2-3.
3. Enter the command **4758132o.exe**; the program files are unpacked and installed into the current directory.
4. If the OS/2 Feature Installer version 1.2 is not on your system, enter the command **fisetup**; the Feature Installer is installed on your system.
5. Enter the command **install**; both the OS/2 Feature Installer and the Netscape Navigator programs are launched.
6. To install the support program, follow the online directions.

   When prompted to choose the directory location of the software, accept the default or choose your own.

How to Install Manually

In the event of problems with the automatic installation process, you can follow this procedure.

1. In an OS/2 window, change to the directory containing the Coprocessor software package you obtained as described in “How to Download and Decipher the Software and FCV” on page 2-3. If this is not a temporary directory, copy the file **4758132o.exe** to a temporary directory and change to that directory.
2. Run the **4758132o.exe** program to unpack the software into your temporary directory.
3. Copy the Support Program software to the working directories. Choose a drive for the working directories and substitute the drive letter for **x** in the following xcopy commands. If you do not already have the working directories, answer **D**, Directory, to the xcopy query about file name or directory name.

```
xcopy *.dll          x:\ibm4758
xcopy *.sys          x:\ibm4758
xcopy csuecnm.cmd    x:\ibm4758
xcopy readme.os2     x:\ibm4758
xcopy csueincl.h     x:\ibm4758\include
xcopy csuesapi.lib   x:\ibm4758\lib
xcopy mac.c          x:\ibm4758\samples
xcopy makefile.os2   x:\ibm4758\samples
xcopy *.clu          x:\ibm4758\clu
xcopy csueclu.exe    x:\ibm4758\clu
xcopy hikm.zip       x:\ibm4758\cnm
```

4. Make a backup copy of your config.sys file Then make the following changes to config.sys, and reboot your system. You can use the OS/2 tedit editor. Substitute the drive letter for **x** in the following xcopy commands.
   - Extend the LIBPATH variable with x:\ibm4758
   - Add a line with device=x:\ibm4758\crypto.sys
   - Add a line with set csudesds=x:\ibm4758\desstore.dat
   - Add a line with set csupkads=x:\ibm4758\pkastore.dat
   - Add a line with set csudesld=x:\ibm4758\deslist
   - Add a line with set csupkald=x:\ibm4758\pkalist
   - If the CLASSPATH variable is present, add ;x:\ibm4758\cnm\HIKM.zip to the CLASSPATH variable
   - If the CLASSPATH variable is not present, add the line CLASSPATH=.;x:\ibm4758\cnm\HIKM.zip

How to Configure the Support Program

The support program does not require additional configuration; however, you may change the environment variables listed in this section. The environment variables determine the location of the key-storage files and the key-record-list files used by CCA applications.

In the list below, *current_directory* represents the the directory within which the application calling the CCA library resides, and *x* represents the current drive.

CSUDESDS Specifies the name of the DES-key-storage file. The default value is *x:\current_directory***DESSTORE.DAT**.

CSUPKADS Specifies the name of the PKA-key-storage file. The default value is *x:\current_directory***PKASTORE.DAT**.

CSUDESLD Specifies the directory where the key-record-list files are created for the DES_Key_Record_List verb. The default value is *x:***KEYDIR**.

CSUPKALD Specifies the directory where the key-record-list files are created for the PKA_Key_Record_List verb. The default value is *x:***PKADIR**.

To change an environment variable, enter on the command line:

set *variable_name = new_directory_and_name*

Note: You can also use **set** within a command file.

The environment variables are read by the CCA library at the beginning of the first request sent to the CCA library after the library is loaded by the operating system. Changes to the variables do not take effect until the operating system unloads and then re-loads the library.

Note: When you initialize key storage using the Cryptographic Node Management Utility, ensure that you specify the directories defined by these variables; see "How to Create or Initialize Key Storage" on page 5-21.

How to Remove the Support Program

Important: 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. Also remember to back up or save any edited role and profile information.

To remove the support program:

1. From the OS/2 window, go to the directory containing the support program software.
2. Enter the command **uninstal**; the OS/2 Feature Installer and Netscape Navigator programs are launched.
3. Follow the online directions; the software is removed from the operational directories. (You must remove the distribution file and the files in the temporary directory in which you unpacked the distribution file.)

Note: If you are unable to remove the IBM 4758 CCA software using the above procedure, you can manually remove the software by following these steps. (These instructions assume the software is on the C: drive, substitute the correct drive letter as required.)

1. Modify the config.sys file:
   a. Remove or comment-out the “device=c:\ibm4758\crypto.sys” device driver statement
   b. Remove “c:\ibm4758” from the LIBPATH statement
   c. Remove or comment-out the four lines:
      - “set csudesds=...”
      - “set csupkads=...”
      - “set csudesld=...”
      - “set csupkald=...”
   d. Reboot your system.
   e. Remove “c:\ibm4758\cnm\JIKM.zip” from the CLASSPATH statement.
2. Erase the files in the “c:\ibm4758” directory structure and remove the directory structure.

How to Install and Remove the Support Program for Windows NT

This section includes a description of the support program system requirements and procedures necessary to install and uninstall the software.

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

NT 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*, SC31-8623.

Software:

Windows NT Version 4.0.

Java runtime environment 1.1.1 through 1.2, available from *http//java.sun.com/products/OV_jdkProduct.html*. This is required to use the Cryptographic Node Management Utility.

Disk Space:

2 MB.

How to Install the Support Program

To install the support program:

1. If a prior release is installed, you must remove the earlier release. Be sure to backup or save the key-storage files, roles, profiles, and any other data or code you are storing in the same directories.
2. Enter the command **4758132w.exe**; the program files are unpacked and installed into the current directory.
3. 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.
4. Adjusting the Windows NT 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 synchronism. The CCA Support

Program presumes that the system clock and the time zone settings have been correctly established.

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

```
SET TZ=EST5EDT,4,1,0,3600,10,-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 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 saving 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

You should also make the TZ setting generally active. To make the TZ setting automatic, go to the Windows NT Control Panel, select **System**, then select **Environment**. In the Variable box, enter **TZ**. In the Value box, enter the *TZ parameters* as defined for the SET TZ statement.

5. 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 Explorer and double-click the IdSelect1.reg file found in the c:\program files\IBM\4758 directory.[1]

How to Remove the Support Program

Important: 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. Also remember to back up or save any edited role and profile information.

To remove the support program:

1. Open the NT **Add/Remove Programs** control panel; **IBM 4758 PCI Cryptographic Coprocessor** is displayed in the list of software.
2. Highlight **IBM 4758 PCI Cryptographic Coprocessor**.
3. Select **Add/Remove...**; you are prompted to confirm your choice.
4. Select **Yes**; the software is removed.

[1] 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.

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 operating system into the Coprocessor.

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

This chapter includes:

- Instructions for using the CLU 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
  - CLU 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 web site Publication page.

Notes:

1. The file locations referenced in this chapter are the default directory paths.
2. 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 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 program. On OS/2 and Windows NT platforms, the CLU program must either be located in the default directory or be included in the path statement. The default directories are:

AIX	/usr/lpp/csuf/clu
OS/2	\IBM4758\CLU
Windows NT	\Program Files\IBM\4758\CLU

On Windows NT you can issue a change directory command that includes a space character by enclosing the parameter in quotes, for example:

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

To run the CLU program, you enter the program name at the command prompt, **CSU**x**CLU** where “x” is different for the three operating systems:

AIX	CSU**F**CLU
OS/2	CSU**E**CLU
Windows NT	CSU**N**CLU

You can provide parameters interactively to the CLU program, or you can include these on the command line input. (Details are provided at “Coprocessor Load Utility Syntax” on page 4-7.) 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. It is strongly recommended to 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 default directory that contains the CLU program. Start CLU with the name appropriate to your operating system. Respond to the prompts as requested.

Determining Coprocessor Software Segment Contents

The Coprocessor has three "segments," 1, 2, and 3. Each segment holds:

- Status
- Software
- A validation public key
- An owner identifier (except for segment 1).

.

Segment	Content
1	"Miniboot," contains diagnostics and code loading controls
2	CP/Q++ control program
3	CCA or another application.

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

```
********** Command ST started. ---- Wed Apr 26 15:02:26 2000

 *** VPD data; PartNum = 69H6479
 *** VPD data;  EC Num = C75554E
 *** VPD data; Ser Num = 41-00027
 *** VPD data;  Description = IBM 4758-001 PCI CRYPTOGRAPHIC COPROCESSOR
 *** ROM Status; PIC ver: 101, ROM ver: 00
 *** ROM Status; INIT: INITIALIZED
 *** ROM Status; SEG2: RUNNABLE , OWNER2: 02
 *** ROM Status; SEG3: RUNNABLE , OWNER3: 02
 *** Page 1 Certified: YES
 *** Segment 1 Image: CCA 1.32 SEGMENT-1
 *** Segment 1 Revision: 132
 *** Segment 2 Image: CCA 1.32 SEGMENT-2
 *** Segment 2 Revision: 132
 *** Segment 3 Image: CCA 1.32 SEGMENT-3
 *** Segment 3 Revision: 132
 *** Query Adapter Status successful ***
   ** Obtain Status
********** Command ST ended. ---- Wed Apr 26 15:03:21 2000
```

Figure 4-1. Typical CLU Status Response

Item **Discussion**

Ser Num The serial number of the Coprocessor, e.g. 41-00027

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

ROM Status, Initialized 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, etc. 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 code 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 initialized 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. "CCA" in the image name means that the contents have been provided as part of the CCA Support Program; be sure to observe the revision level.

Changing Software Segment Contents

The CLU program provided with the CCA Support Program Release 1.32 can process .CLU files from the current and previous releases. However, special instructions are required to use the files from earlier releases. The instructions in this chapter apply to files provided with Release 1.32.

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 release levels except with specific instructions.

Start the CLU program 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**.
- Enter the CLU file name as indicated in the README file.

Repeat as required so that the proper software remains in 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 model in use,[1] issue one of these commands:

- For a Model 001:

```
CSUxCLU ######## VA STPPUVAL.CLU
```

- For a Model 013:

```
CSUxCLU ######## VA STL1M13V.CLU
```

The README file describes the Image Names that you should observe.

[1] You can refer to the IBM 4758 product website (*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.

How to Unload Coprocessor Software and Zeroize CCA

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 segments status is set to “UNOWNED.”

You will need to 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 program, **CSU**x**CLU**
- 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.

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 vallidating 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 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
- CLU return codes.

Coprocessor Memory Segments

Coprocessor memory segments are organized as follows:

Segment_0 “BIOS”

The BIOS 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:

- Administers the replacement of software already loaded to Segment_1.
- Administers the loading of data and software to segments two and three.
- Is loaded at the factory, but can be replaced using the CLU.

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 segments 2 and 3 is a two-step process.

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 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 segment[2].

   The Package Load (PL) command can be used to process a file that contains several commands and associated data for the CLU program. Files intended for use with the PL command incorporate Load and/or Reload commands.

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.
- 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:

- Obtain the release level and the status of software currently installed to 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.

---

[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.

To invoke the utility:

1. Log on as required by your operating system.

   Note to AIX Users: The load process requires root-level authority; it must be performed by a system administrator with that authority, or a member of the "csufadmin" group.

2. Go to the command line.

3. Change directory to the directory containing the CLU files. The default directories are:

AIX	*usr/lpp/csuf/clu*
OS/2	*\IBM4758\clu*
NT	*\Program Files\IBM\4758\clu*

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

AIX	**csufclu**
OS/2	**csueclu**
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* [*adpt#*] [*datafile_name* [*-q*]]]]

Example: The **VA** command that validates the status of the Coprocessor requires a "class-key certificate file." The IBM 4758 product website (*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 website. 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 ######## 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."

cmd — A two-letter abbreviation representing the command to be executed. These abbreviations are listed below.[3]

[3] Prior releases of the CLU utility used additional commands to control ownership and code loading into the Coprocessor (commands R1, E2, L2, R2, S2, E3, L3, R3, and S3). With the current release, these commands are inferred from information

adpt# Provides the adapter number as established by the device driver. This parameter defaults to zero. Adapters are designated to the device driver as numbers 0, 1, ..., n. 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 adpt#.

datafile_name Identifies the datafile (drive, directory, and filename) used for the operation requested.

- For software loads and re-loads it is the filename of the software image you are loading into the Coprocessor.
- When obtaining Coprocessor status with the VA command, it is the certificate file used to validate the Coprocessor response: **stppuval.clu** or **stl1m13v.clu**.

-q You can use this "quiet" parameter to suppress 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:

ST: Obtain Status

Obtain the status of loaded software and the release level of other components and then store in the utility logfile.

VA: Obtain Status with Validation

Obtain 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 Load

Process a series of the commands as directed by the contents of the datafile to establish segment ownership and to load or reload segment software.

RS: Reset Coprocessor

Use this command to reset 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 loose synchronization. You must end all host-system software processes that are operating with the

---

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. The Release 1.31 CLU program also does not include the EC command to recover the SKA Certificate. IBM dos not intend to support processing of the SKA certificate.

The Release 1.31 and 1.32 CLU program will not process .CLU files from earlier releases. If you might need to setup a Coprocessor with the older code, be sure to retain the .CLU files and the CLU program from the earlier release.

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 written to the display. (By default, the utility returns prompts and messages to the display.)

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 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 datafile is required with this command.

7 The datafile 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.
“OS/2 Requirements” on page 3-6.
“NT Requirements” on page 3-10.

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 utility:

- On AIX, enter **csufcnm** on the command line.
- On OS/2 systems:
  - Change directory to **\ibm4758\cnm**
  - Enter **csuecnm** on the command line.
- On Windows NT systems:
  - Change directory to **\program files\ibm\4758\cnm**
  - Enter **csuncnm** on the command line.

The Cryptographic Node Management Utility logo and then the main panel are displayed.

Cryptographic Node Initialization Utility Overview

The Cryptographic Node Initialization (CNI) utility executes 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 execute the Cryptographic Node Management Utility functions necessary to set up a node; for example, it can be used to load the Coprocessor master keys.

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, etc.) 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 5-23.

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 5-9 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 for your specific environment.

How to Establish a Test Node

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

Important: Because many sensitive operations are permitted unrestricted use under this scenario, the resulting cryptographic node should not be considered secure.

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. 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.
3. 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 5-10.

   Finish this task by selecting ***Cancel***.
4. 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 obtained when you used your IBM Customer Number and License Key to unwrap the encrypted FCV file downloaded from the Web. FCVs usually have file names such as "CCA5200.FCV" and can be found by using the file search utility available with your operating system.
5. 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 the Master Key" on page 5-16.
6. *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: 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 5-23.

   a. Create a Cryptographic Node Initialization (CNI) 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 5-23.

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 have 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:

- 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 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
- Clearing of all retained and registered keys
- Clearing all 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 is a signed value provided by IBM and used to enable the CCA application in the Coprocessor to provide a level of cryptographic service consistent with (applicable) import and export regulations. Use the utility to load into the Coprocessor the function-control vector you downloaded through the Internet. See "How to Download and Decipher the Software and FCV" on page 2-3.

The FCV file has a name of the form "CCA520x.FCV," where x = 0, 1, 2, or 3. You can locate this file with 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 allows you to indicate directory paths for the files you create with the utility. The utility generally *does not* use the paths that you indicate, but you may find it a useful place to record where you intend to keep 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 5-15.
- *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 un-authenticated 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 un-authenticated 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.

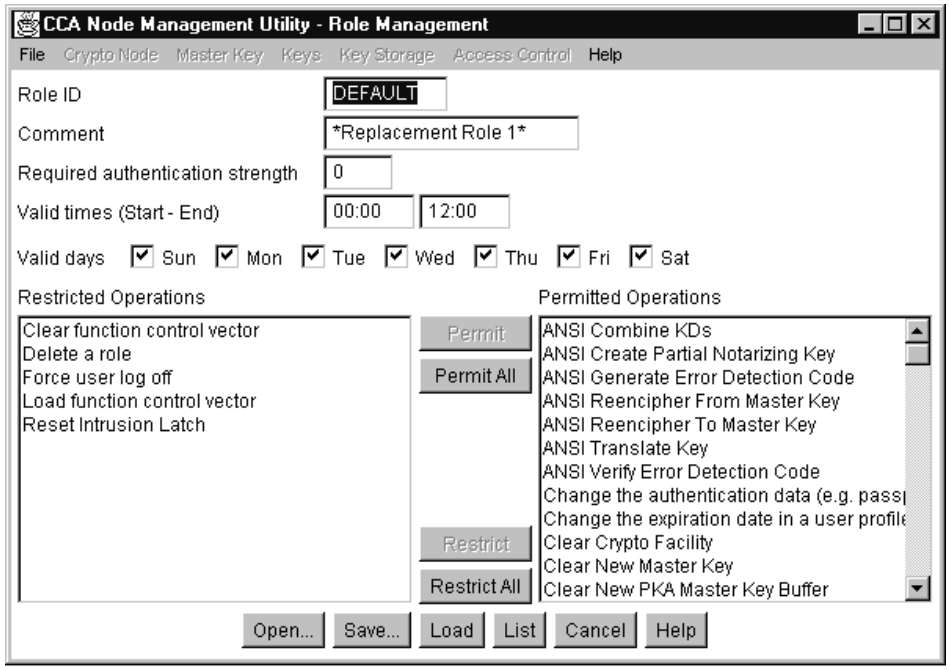

Figure 5-1. The Role Definition Panel

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*, SC31-8609.

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 execute the verb.

For more information about CCA verb calls and commands, refer to the *IBM 4758 CCA Basic Services Reference and Guide*, SC31-8609. 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 User Profile Definition panel; see Figure 5-2 on page 5-13.
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. (This entry field, added in Release 1.31, is not shown in Figure 5-2 on page 5-13.) The utility will set this by default to 90 days from the current 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; see Expiration Date below.

   Comment
   An optional character string.

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

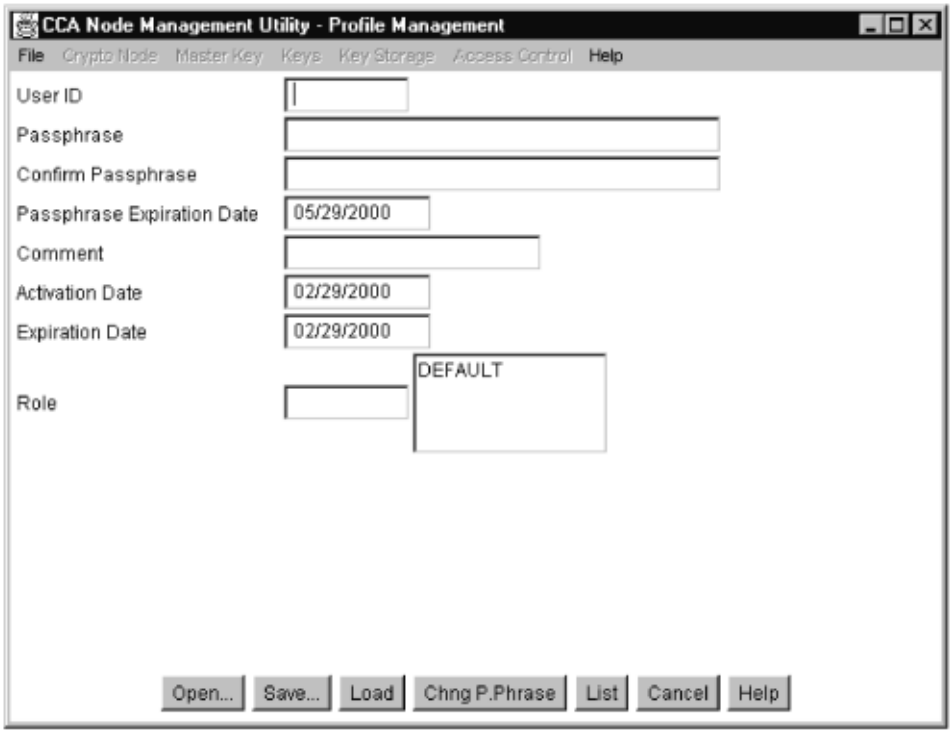

Figure 5-2. The User Profile Definition Panel

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.

Note: 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 assymetric 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.

For information about checking the contents of these registers, see “How to Obtain Status Information” on page 5-8.

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 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...***; you are prompted to verify the command.
2. Select **Yes**; the Coprocessor generates and sets a master key.

Important: When you set or auto-set a master key, you must re-encipher all keys enciphered under the former key; see “How to Reencipher Stored Keys” on page 5-21.

How to Load a New Master Key from Parts

To set a new master key into the Coprocessor, load the first, any middle, and last master 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 on page 5-17.

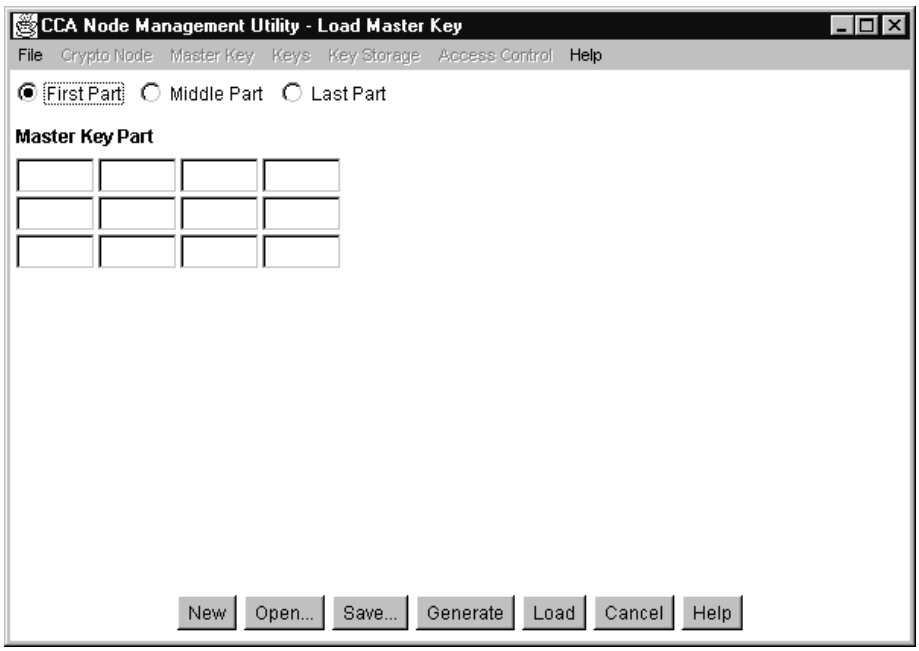

Figure 5-3. The Load Master Key Panel

2. Select the radio button for the key_part you are editing (first, middle, or last).
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, re-encipher the keys currently in storage; see “How to Reencipher Stored Keys” on page 5-21.

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 copy 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* publication. 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 need to 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 $share_i$, $1 \geq i \geq n$. 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, etc.

   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 one 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').

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.

   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.

5. 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***.

   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.

   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.

6. 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.

7. 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.

8. At the source node have authorized individuals sign on to the role that permits each of them to obtain their share. 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 and 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***.

9. At the target node have authorized individuals sign on to the role that permits each of them to install their 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***.

10. 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
- Re-encipher 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). The ODM locations are defined by your system administrator; see "How to Configure the Support Program" on page 3-3 and the use of the ODMGET and CSUFKEYS utilities.

   Note to OS/2 Users: The location you set for key storage *must* match the location defined by the environment variables described in the section "How to Configure the Support Program" on page 3-7.

   Note to Windows NT 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 NT 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 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 key-encrypting key 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 key-encrypting key.

Note: The Cryptographic Node Management Utility supports DES KEKs only 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:

1. From the **Keys** pull-down menu, select ***Primary DES Key-Encrypting Keys***; the Primary DES Key-Encrypting Keys panel is displayed.
2. Select the radio button for the first key_part.
3. Enter data in the **Key Part** by using one of the following processes:
   - 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 "Key Part" fields; each field accepts four hexadecimal digits.
4. Select a control_vector for the key:
   - To use the default control_vector for importer keys or exporter keys, select the appropriate radio button.

- To use a custom control_vector, select the **Custom** radio button and enter a control_vector. For information about control_vectors, refer to Appendix C of the *IBM 4758 CCA Basic Services Reference and Guide*.

5. Enter a key label, a unique name, to identify the key in storage. You must use the same key label name for all three parts of the KEK that you generate.
6. Select **Load** to load the key part into the Coprocessor; select **Save...** to save it to disk.
7. Save or load the remaining key parts by following Step 2 on page 5-22 to Step 6. 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 Node Initialization Utility (CNI), 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 CNI Editor panel is displayed. See Figure 5-4.

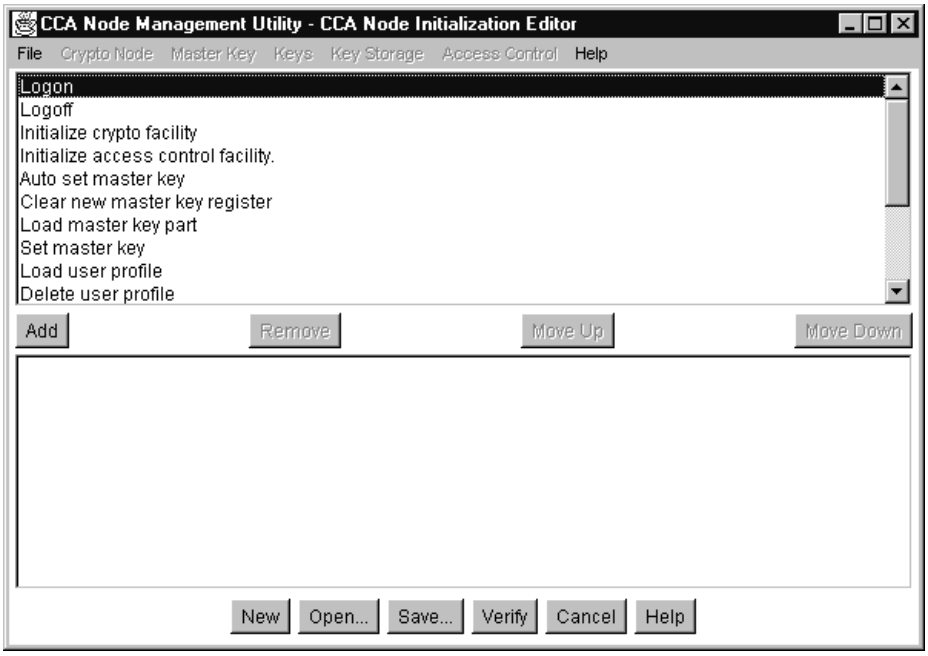

Figure 5-4. The CNI Editor Panel

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 filename 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, execute the list using the Cryptographic Node Initialization Utility:
    - On AIX systems, enter **csufcni** *listfile_name* on the command line.
    - On OS/2 systems:
      – Change directory to **\ibm4758\cnm**
      – Enter **csuecni** *listfile_name* on the command line.
    - On Windows NT 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. The observations are found under these headings:

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

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 code from IBM's website. Except for patched modules that may be offered from time to time on the website, the downloaded code package is an executable program with an encrypted payload. Use your IBM Customer Number and the License Key that you obtained from IBM with the downloaded program to have the payload decrypted and installed into the distribution directories. See Chapter 2, “Ordering and Obtaining Coprocessor 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 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.

Access Controls

The access-control system and the grouping of permissible commands[1] 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.

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

The following sections describe:

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

Locking the Access-Control System

For secure operation after initializing processes, consider “locking” the access-control system. You can lock the access-control system by removing any profile that would allow use of the **INIT-AC** keyword on the Access_Control_Initialization verb (CSUAACI), thereby preventing further changes to the access controls.

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.

Passphrase Considerations

The passphrase used to authenticate access to a profile is not communicated out of the SAPI layer during logon. 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, you use the **PROTECTD** keyword. This causes the passphrase to be encrypted within the SAPI DLL layer before it is communicated to the Coprocessor. You remain exposed to a rogue process within your system being able to determine passphrase values.

In the current implementation, if a role has 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.

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.

Roles and Profiles

The access-control system, which is discussed in the opening pages of Chapter 2, “Ordering and Obtaining Coprocessor Software,” permits users to define roles and profiles as suits their operation and security needs. Roles and profiles you can consider include:

Table 6-1 (Page 1 of 3). Example Roles

Setup	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.

Table 6-1 (Page 2 of 3). Example Roles

Administrator	You can establish an Administrator role(s) with extensive supervisory capabilities. The administrative roles could be permitted to: • Change the passphrase of any profile and reset the failure count of any profile (Access_Control_Initialization verb). 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 would 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 passphrases. You can address this problem in these ways: • 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. **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.
SO1, Security Officer 1	Security Officer 1's role(s) could be permitted to: • Randomly generate a master key • Import a key-encrypting key.
SO2, Security Officer 2	Security Officer 2's role(s) could be permitted to: • Set a master key • Import keys. **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.
Default	You must have a Default role. When a host thread is not logged on, requests from such a process thread are accepted based on the permissions set in the default role. You should enable only those control points 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).

Table 6-1 (Page 3 of 3). Example Roles

Application User$_n$	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. 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.

In all cases, only those control points 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 that a rogue process not be able to make use of the encrypted key.

With CCA, you can often make use of a unique capability afforded through the CCA control vector and command architecture. CCA permits DES keys to have asymmetric properties. Using MAC/MACVER, ENCIPHER/DECIPHER, IMPORTER/EXPORTER, PINGEN/PINVER, and IPINENC/OPINENC keys, 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 only having the key in the MACVER class is unable to create a MAC with the key. Thus, data recipients (who also 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.

ENCIPHER/DECIPHER
You can distribute the ability to reverse a DES ciphering process between nodes.

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.

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.)

Note however that the key-part information flows in the clear through your host system. In some cases you may view this as an unacceptable 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 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 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.)

You should have a concern for the export of keys from your system. Take special care in the enablement of the three key-export verbs: Data_Key_Export, Key_Export, and PKA_Symmetric_Key_Export. Note especially that PKA_Symmetric_Key_Export permits 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.

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 request can be disallowed through the access-control system. This is a potentially unsecure 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 key-encrypting key can be created. This request can be disallowed through the access-control system. This is a potentially unsecure 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 Clear_Key_Import	Eight bytes of clear information can be accepted to be returned as an encrypted DATA key. This request 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.
CSNBKPI Key_Part_Import	This verb requires use of two commands (using the **FIRST**, **MIDDLE**, or **LAST** keywords) to complete the establishment of a productive key of any type. The key information is passed in the clear. These requests 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** keyword. 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 *pre-XOR* technique.
CSNBMKP Master_Key_Process	Use of the **FIRST**, **MIDDLE**, and **LAST** keywords employs clear data to establish the value of a master key. These requests 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 *master-key cloning* process.

PIN Data

A Personal Identification Number (PIN) is generally passed across the 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 request 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 request 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 well 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 the current release, the version of CP/Q++ distributed with CCA and using segment 2 owner identifier 2 does not support the use of the RS-232 serial interface. No information from CCA can pass over this interface.

Master-Key Cloning

If master-key cloning will be employed, 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 6-2.

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
- Additional application program termination procedure in AIX
- A sample routine written in the C programming language.

  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 verbs[1]. The OS/2 and NT 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, OS/2, and NT 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
OS/2	\IBM4758\include\csueincl.h
NT	\Program Files\IBM\4758\include\csunincl.h

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

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

AIX	`#include <csufincl.h>`
OS/2	`#include "csueincl.h"`
NT	`#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. Code null address pointers for these parameters. Or 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. These files reside in directory: The file and its default distribution location is:

AIX	*/usr/lpp/csuf/samples/c*
OS/2	*\IBM4758\samples*
NT	*\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 Microsoft or other vendors.

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

AIX	*/usr/lib/libcsufsapi.a*
OS/2	*\IBM4758\lib\csuesapi.lib*
NT	*\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, xlc_r.

Additional Application Program Termination Procedure in AIX

In order to allow the CCA support program to properly cleanup resources in AIX, your application must call the function "CCA_CLOSE_RESOURCES" prior to terminating or prior to explicitly discarding the CCA shared libraries. See the CCA application header file (csufincl.h) for the function prototype. This function returns a boolean to indicate success (true) or failure (false).

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.

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.

Table 7-1. Verbs Called by the Sample Routine

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 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 may be     */
/* used. Any functionally equivalent program may be used.            */
/*                                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
/* This example program:                                             */
/*                                                                   */
/* 1) Calls the Key Generate verb (CSNBKGN) to creates a MAC 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 message authentication code (MAC)    */
/*    on the supplied text string (INPUT_TEXT).                      */
/*                                                                   */
/* 3) Calls the MAC Verify verb (CSNBMVR) to verify the message      */
/*    authentication code (MAC) for the same text string, using the  */
/*    MACVER key token created in step 1.                            */
/*                                                                   */
/*********************************************************************/
```

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

```
#include <stdio.h>
#include <string.h>

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

/* Defines */

#define INPUT_TEXT              "abcdefhgijklmn0987654321"
#define MAC_PROCESSING_RULE     "X9.9-1  "
#define SEGMENT_FLAG            "ONLY    "
#define MAC_LENGTH              "HEX-9   "
#define MAC_BFR_LENGTH          10
#define KEY_FORM                "OPOP"
#define KEY_LENGTH              "SINGLE  "
#define KEY_TYPE_1              "MAC     "
#define KEY_TYPE_2              "MACVER  "

void main()
{
  static long          return_code;
  static long          reason_code;
  static long          exit_data_length;
  static unsigned char exit_data[4];
  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 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 long          text_length;
  static unsigned char text[26];
  static long          rule_array_count;
  static unsigned char rule_array[8][8];
  static unsigned char chaining_vector[18];
  static unsigned char mac_value[MAC_BFR_LENGTH];

  /* Print a banner */

  printf("Cryptographic Coprocessor Support Program example program.\n");
```

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

```
memset (mac_value, 0x00, sizeof(mac_value)); /* Clear the mac value.       */

memcpy (key_form,        KEY_FORM,   4);     /* Set up initial values.     */
memcpy (key_length,      KEY_LENGTH, 8);
memcpy (mac_key_type,    KEY_TYPE_1, 8);
memcpy (macver_key_type, KEY_TYPE_2, 8);

/* Generate a key. */

CSNBKGN(&return_code,
        &reason_code,
        &exit_data_length,
        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 code.  Terminate if there is an error.          */

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

  printf ("Key Generate Failed\n");            /* Print failing verb.      */
  printf ("Return_code = %ld\n", return_code); /* Print return code.       */
  printf ("Reason_code = %ld\n", reason_code); /* Print reason code.       */
  return;
}

else {                                         /* No error occurred.       */

  printf ("Key Generate Successful\n");
```

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

```
memcpy (text, INPUT_TEXT, sizeof (INPUT_TEXT) - 1); /* Get the input text*/

text_length = sizeof (INPUT_TEXT) - 1;       /* Set the text length.     */

rule_array_count = 3;                        /* Set rule array           */

memcpy (rule_array[0], MAC_PROCESSING_RULE, 8);
memcpy (rule_array[1], SEGMENT_FLAG,        8);
memcpy (rule_array[2], MAC_LENGTH,          8);

memset (chaining_vector, 0x00, 18);         /* Clear the chaining vector.*/

/* Call MAC_Generate. */

CSNBMGN (&return_code,
         &reason_code,
         &exit_data_length,
         exit_data,
         mac_key_id,
         &text_length,
         text,
         &rule_array_count,
         &rule_array[0][0],
         chaining_vector,
         mac_value);

/* Check the return/reason code.  Terminate if there is an error.        */

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

  printf ("MAC Generate Failed\n");            /* Print failing verb     */
  printf ("Return_code = %ld\n", return_code); /* Print return code      */
  printf ("Reason_code = %ld\n", reason_code); /* Print reason code      */
  return;
}

else {                                         /* No error occurred      */

  printf ("MAC Generate Successful\n");
  printf ("MAC_value = %s\n", mac_value);      /* Print MAC value        */
```

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

```
      /* Set the rule array for the MAC Verify. Use the default MAC         */
      /* Ciphering Method and Segmenting Control.                           */

      rule_array_count = 1;                        /* Set the rule array     */
      memcpy (rule_array[0], MAC_LENGTH, 8);

      /* Call MAC_Verify         */

      CSNBMVR (&return_code,
               &reason_code,
               &exit_data_length,
               exit_data,
               macver_key_id,
               &text_length,
               text,
               &rule_array_count,
               &rule_array[0][0],
               chaining_vector,
               mac_value);

      /* Check the return/reason code.  Terminate if there is an error.     */

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

        printf ("MAC Verify Failed\n");             /* Print failing verb.  */
        printf ("Return_code = %ld\n",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

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, a 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*, SC31-8609.

Table A-1 (Page 1 of 3). 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'	Re-Encipher to Master Key	Key_Import	CSNBKIM	O
X'0013'	Re-Encipher from Master Key	Key_Export	CSNBKEX	O
X'0018'	Load First Master Key Part	Master_Key_Process†	CSNBMKP	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	Master_Key_Process†	CSNBMKP	O, SUP
X'0033'	Clear Old Master Key Register	Master_Key_Process†	CSNBMKP	O, SUP
X'0040'	Generate Diversified Key	Diversified_Key_Generate	CSNBDKG	O
X'008C'	Generate Key Set	Key_Generate††	CSNBKGN	O
X'008E'	Generate Key	Key_Generate†† Random_Number_Generate	CSNBKGN CSNBRNG	R
X'0090'	Re-Encipher to Current Master Key	Key_Token_Change	CSNBKTC	R
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'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

The following codes are used in this table:

AA This command is always authorized.
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.
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, as determined by the nature of the key affected and the action being performed upon it.

Table A-1 (Page 2 of 3). Supported CCA Commands

Code	Command Name	Verb Name	Entry	Usage
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	CSNBCKI	O
X'00D7'	Generate Key Set Extended	Key_Generate††	CSNBKGN	SC, SUP
X'00DB'	Replicate Key	Key_Generate††	CSNBKGN	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	CSNDPKG	O, SUP
X'0105'	PKCS-1.2 Symmetric Key Export	PKA_Symmetric_Key_Export	CSNDSYX	SC
X'0106'	PKCS-1.2 PKA Symmetric Key Import	PKA_Symmetric_Key_Import†	CSNDSYI	O
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
X'010D'	PKA Symmetric Key Generate	PKA_Symmetric_Key_Generate†	CSNDSYG	SC
X'010E'	NL-EPP-5Symmetric 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	SUP
X'0111'	Reinitialize Device	Cryptographic_Facility_Control†	CSUACFC	SUP
X'0112'	Initialize Access-Control System	Access_Control_Initialization†	CSUAACI	SUP
X'0113'	Change User Profile Expiration Date	Access_Control_Initialization†	CSUAACI	SUP
X'0114'	Change User Profile Authentication Data	Access_Control_Initialization†	CSUAACI	SUP
X'0115'	Reset User Profile Logon-Attempt-Failure Count	Access_Control_Initialization†	CSUAACI	SUP
X'0116'	Read Public Access-Control Information	Access_Control_Maintenance†	CSUAACM	O
X'0117'	Delete User Profile	Access_Control_Maintenance†	CSUAACM	SUP
X'0118'	Delete Role	Access_Control_Maintenance†	CSUAACM	SUP
X'0119'	Load Function-Control Vector	Cryptographic_Facility_Control†	CSUACFC	SUP
X'011A'	Clear Function-Control Vector	Cryptographic_Facility_Control†	CSUACFC	NR
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'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

The following codes are used in this table:

AA This command is always authorized.
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.
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, as determined by the nature of the key affected and the action being performed upon it.

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

Code	Command Name	Verb Name	Entry	Usage
X'0204'	PKA Clear Key Generate	PKA_Key_Generate†	CSNDPKG	O, SUP
X'0211' through X'021F'	Clone-info (share) Obtain	Master_Key_Distribution†	CSNBMKP	O, SUP
X'0221' through X'022F'	Clone-info (share) Install	Master_Key_Distribution†	CSNBMKP	O, SUP
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 PKA Symmetric Key Import	PKA_Symmetric_Key_Import†	CSNDSYI	O

The following codes are used in this table:

AA This command is always authorized.
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.
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, as determined by the nature of the key affected and the action being performed upon it.

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.***:*** The following table lists the access-control commands furnished to the initial-DEFAULT role:

Table B-1. Initial-DEFAULT Role Commands

Code	Command Name
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. 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 Coprocessor accessing code 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 8xxxxxxx (in hexadecimal). The most likely codes that may be encountered are described in Table D-1 on page D-2. If you encounter codes of the form X'8340xxxx' or X'8440xxxx', contact the IBM 4758 Support organization for advice via the question form on the IBM 4758 product website (*http://.www.ibm.com/security/cryptocards*).

Table D-1. Device Driver Error Codes in the Class X'8xxxxxxx'

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.
8040FFDB	Xray tamper	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.
8040FFFE	Battery warning	Battery power is marginal. The battery changing procedure described in the IBM 4758 Installation Manual should be followed to replace the batteries.
804xxxxx (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 problems persists, contact IBM 4758 Support via the website.
8340xxxx	Miniboot-0 codes	This class of return code arises from the lowest-level of reset testing.
8340038F	Random number generation fault	Continuos 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.

Appendix E. 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 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 product website at *http://.www.ibm.com/security/cryptocards*.

- Feature Code 4374 identifies the AIX workstation software.
- Feature Code 4372 identifies the OS/2 workstation software.
- Feature Code 4376 identifies the Windows NT workstation software.

The strength of cryptographic services provided by the product is determined by the function-control vector accompanying it. Refer to the product website for the latest listing of function-control vectors available.

The CCA Cryptographic Coprocessor Support Program must be used in accordance with the IBM System Programs 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	SecureWay

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

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

List of Abbreviations and Acronyms

ANSI	american national standards institute
AIX	advanced interactive executive (operating system)
API	application program interface
ASCII	american national standard code for information interchange
C	celsius
CA	certification authority
CBC	cipher block chain
CCA	common cryptographic architecture
CDMF	commercial data masking facility
CDSA	cryptographic data security architecture
CLU	Coprocessor load utility
CP/Q++	control program/q with 4758 extensions
CPU	central processing unit
CSP	cryptographic service provider
CV	control vector
DEA	data encryption algorithm
DES	data encryption standard
DMA	direct memory access
ECB	electronic codebook
EPROM	erasable programmable read only memory
FCC	federal communications commission
FCV	function-control vector
FIPS	federal information processing standard
IBM	international business machines
ICSF	integrated cryptographic service facility
I/O	input/output
IPL	initial program load
ISO	international organization for standardization
KEK	key encrypting key
LU	logical unit
MB	megabyte
MAC	message authentication code
MD5	message digest 5 (hashing algorithm)
MDC	modification detection code
MHz	megahertz
NIU	node initialization utility
ODM	object data manager
OEM	original equipment manufacturer
OS/2	operating system/2
PC	personal computer
PCI	peripheral component interconnect
PDD	physical device driver
PDF	portable document format
PIN	personal identification number
PKA	public key algorithm
PKCS	public key cryptography standard
POST	power on self test
PPD	program proprietary data
RAM	random access memory
RNG	random number generator
ROM	read only memory
RSA	rivest, shamir, and adleman (algorithm)
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 42 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-ORd 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 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. (I) 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) (I) (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

E

F

H

I

K

L

M

N

O

P

R

S

T

U

V

Z

CCA Release 1.32 IBM 4758 Models 1 & 13

Printed in U.S.A.