z/OS
Integrated Cryptographic Service Facility

System Programmer's Guide

SA22-7520-04

z/OS
Integrated Cryptographic Service Facility

System Programmer's Guide

SA22-7520-04

Note!
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 201.

Fifth Edition (June 2003)

This is a major revision of SA22-7520-03.

This edition applies to Version 1 Release 4 of z/OS (5694-A01) and Version 1 Release 4 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this document

This document supports z/OS (5694-A01) and z/OS.e (5655-G52). It describes how to initialize, customize, operate, and diagnose the z/OS Integrated Cryptographic Service Facility (ICSF). The z/OS Cryptographic Services includes the following components:

- z/OS Integrated Cryptographic Service Facility (ICSF)
- z/OS Open Cryptographic Services Facility (OCSF)

ICSF is a software element of z/OS that works with the hardware cryptographic feature and the Security Server (RACF) to provide secure, high-speed cryptographic services. ICSF provides the application programming interfaces by which applications request the cryptographic services.

Who Should Use This document

This document is intended for the system programmer. It describes the following tasks that a system programmer might perform:

- Programming installation options, installation-defined callable services, and installation exits
- Creating the data sets that ICSF uses
- Migrating the system from the Cryptographic Unit Support Program (CUSP) and Programmed Cryptographic Facility (PCF) to ICSF
- Migrating to z/OS ICSF
- Migrating from the IBM Network Security Processor Support Program (hereafter called 4753-HSP) to ICSF
- Starting and stopping ICSF
- Checking event recording
- Planning for security and performance considerations
- Debugging and recovering from problems

Defining and writing installation-defined callable services and installation exit routines is intended to be accomplished primarily by experienced system programmers. This information assumes that the reader has an advanced knowledge of z/OS.

How to Use This document

The information in this document is divided into descriptions of the following tasks:

- Introducing ICSF
  - Chapter 1, “Introduction to z/OS ICSF”, on page 1, introduces the cryptographic key data set (CKDS) and provides basic information about running PCF and 4753-HSP applications on ICSF and preparing for installation.
- Initializing ICSF
  - Chapter 2, “Installation, Initialization, and Customization”, on page 9, describes how to customize SYS1.PARMLIB, create the CKDS, the PKDS, the installations options data set, the startup procedure, and provide access to the ICSF panels. It also explains how to setup for SMP/E electronic delivery, change the parameters in the installation options data set after the first start and introduces installation exits.

  - Chapter 3, “Migration from PCF to z/OS ICSF”, on page 33, describes how to migrate application programs and cryptographic key data set information to z/OS ICSF from the IBM cryptographic products CUSP/PCF.
  - Chapter 4, “Migration from Previous Releases of ICSF”, on page 51, describes migration to z/OS ICSF from previous releases of ICSF.
  - Chapter 5, “Compatibility and Coexistence of 4753-HSP and ICSF”, on page 71, gives a brief overview of migrating 4753-HSP key storage to ICSF CKDS.
- Customizing ICSF
  - Chapter 6, “Installation-Defined Callable Services”, on page 73 gives information that an experienced system programmer can use to write installation-defined callable services. It also explains how to define these callable services to ICSF, and how to write service stubs to access them.
  - Chapter 7, “Installation Exits”, on page 83, describes the ICSF installation exits you can use to customize ICSF.
- Operating ICSF
  - Chapter 8, “Operating ICSF”, on page 129, describes how to start, modify, and stop ICSF and other operating considerations.
  - “Event Recording” on page 136, describes ICSF event recording on the Security Console and SMF.
- Planning ICSF
  - “Security Considerations” on page 142, describes methods you can use to protect ICSF resources.
- Diagnosing ICSF
  - “Debugging Aids” on page 144, describes the use of component trace and Interactive Problem Control System (IPCS) to debug ICSF.
  - Appendix A, “Diagnosis Reference Information”, on page 149, maps the cryptographic key data set and the cryptographic communication vector tables as reference information for use in debugging. This appendix also maps DES and PKA key tokens.
  - Appendix B, “Installing the CICS-ICSF Attachment Facility”, on page 181, defines steps to install the CICS-ICSF Attachment Facility.
  - Appendix C, “Helpful Hints for ICSF First Time Startup”, on page 187, defines helpful hints and that you may encounter when starting ICSF for the first time.
  - Appendix E, “Running HCR7708 on a IBM @server zSeries 990”, on page 195 describes processing and functionality support for this environment.
  - Appendix F, “Accessibility”, on page 199 contains information on accessibility features in z/OS.
  - “Notices” on page 201 contains information on notices, programming interface information and trademarks.

Where to Find More Information

For more information about other ICSF documents, see Figure 1 on page xvi.

This document also refers to the following publications:

- *IBM ES/3090 Processor Complex Recovery Guide*, SC38-0070
- *z/OS and z/OS.e Planning for Installation*, GA22-7504
- *z/OS MVS IPCS User’s Guide*, SA22-7596
- *z/OS MVS System Codes*, SA22-7626

- *z/OS MVS System Management Facilities (SMF)*, SA22-7630
- *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614
- *z/OS MVS Initialization and Tuning Guide*, SA22-7591
- *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- *MVS Batch Local Shared Resources*, GC28-1469
- *MVS/ESA VSAM Administration Guide*, SC26-4518
- *MVS/DFP Managing VSAM Data Sets*, SC26-4568
- *IBM Transaction Security System: General Information Manual and Planning Guide*, GA34-2137
- *IBM Transaction Security System: Concepts and Programming Guide: Volume I, Access Controls and DES Cryptography*, GC31-3937
- *IBM Transaction Security System: Basic CCA Cryptographic Services*, SA34-2362
- *IBM Transaction Security System: Concepts and Programming Guide: Volume II, Public-Key Cryptography*, GC31-2889
- *IBM Distributed Key Management System, Installation and Customization Guide*, GG24-4406
- *OS/VS1 and OS/VS2 MVS Cryptographic Unit Support: Installation Manual*, SC28-1016
- *OS/VS1 and OS/VS2 MVS Programmed Cryptographic Facility*, SC28-0956
- *CICS Customization Guide*, SC34-5989
- *CICS Resource Definition Guide*, SC34-5990

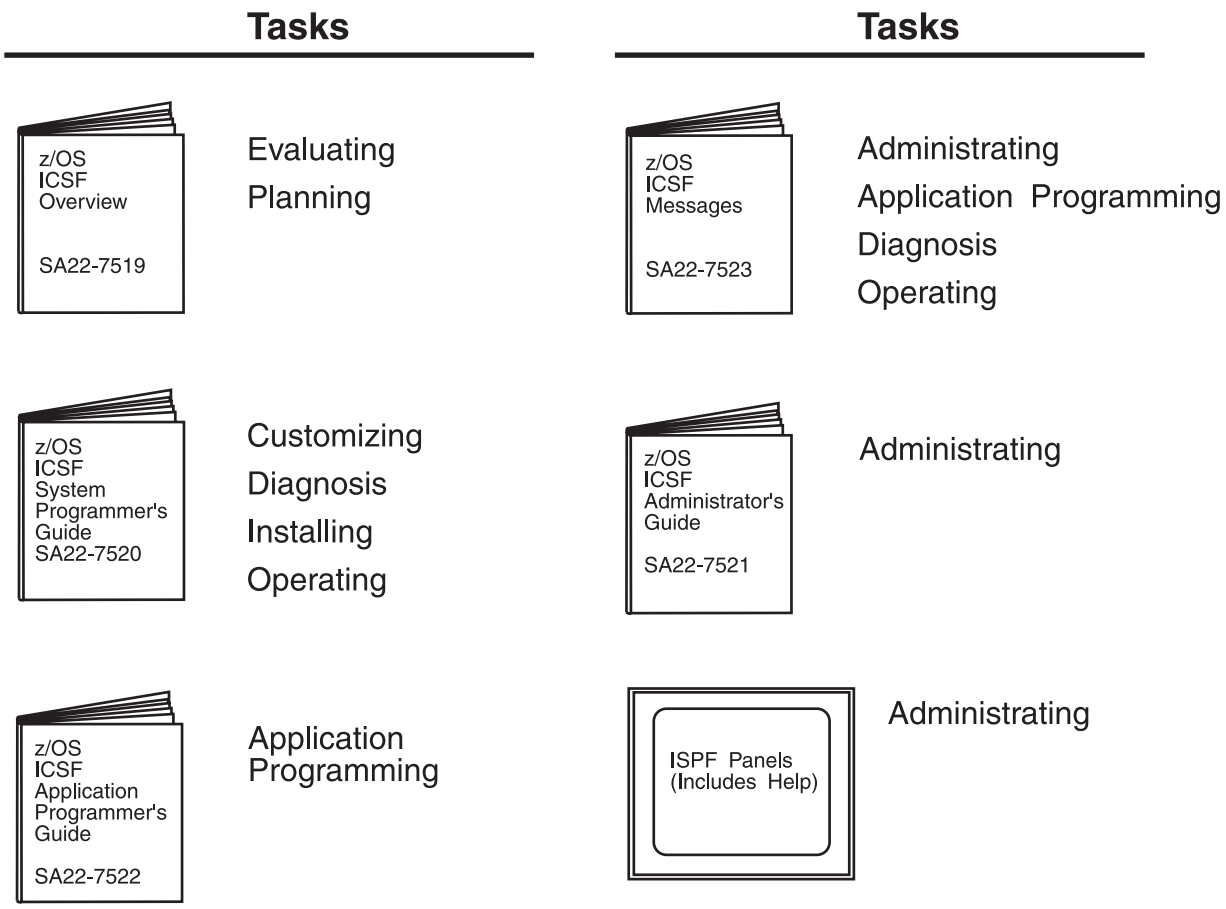


Figure 1. The z/OS ICSF Library

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:
http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/ or from anywhere in z/OS or

z/OS.e where you can access a TSO/E command line (for example, TSO/E prompt, ISPF, z/OS UNIX System Services running OMVS).

The LookAt Web site also features a mobile edition of LookAt for devices such as Pocket PCs, Palm OS, or Linux-based handhelds. So, if you have a handheld device with wireless access and an Internet browser, you can now access LookAt message information from almost anywhere.

To use LookAt as a TSO/E command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO/E from a disk on your *z/OS Collection* (SK3T-4269) or from the LookAt Web site's **Download** link.

Accessing z/OS™ licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

Printed licensed documents are not available from IBM.

To print licensed documents, you can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link .

Do You Have Problems, Comments, or Suggestions?

Your suggestions and ideas can contribute to the quality and the usability of this document. If you have problems using this document, or if you have suggestions for improving it, complete and mail the Reader's Comment Form found at the back of the document.

1. z/OS.e™ customers received a Memo to Licensees, (GI10-0684) that includes this key code.

Summary of changes

Summary of changes
for SA22-7520-04
z/OS Version 1 Release 4

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-03, which supports z/OS Version 1 Release 3.

New information

- Information is added to indicate this document supports z/OS.e and the IBM @server zSeries 800.
- Support for the IBM @server zSeries 990 server has been added. If you are running ICSF in this environment, refer to Appendix E, "Running HCR7708 on a IBM @server zSeries 990", on page 195.
- Callable services
  - Symmetric Key Decipher (CSNBSYD1) - ALET support
  - Symmetric Key Encipher (CSNBSYE1) - ALET support
- Access Control Points
  - Data Key Export - Unrestricted
  - Data Key Import - Unrestricted
  - Key Export - Unrestricted
  - Key Import - Unrestricted

Changed information

- Callable services
  - Callable services (Usage notes section) have been enhanced to include a table which lists the required hardware (by server) and restrictions for the callable service.
  - Encrypted PIN Verify (CSNBPVR) - *rule_array* enhanced to support the VISAPVV4 keyword.
  - MAC Generate (CSNBMGN) and MAC Verify (CSNBMVR) has been enhanced to support longer text on a PCI Cryptographic Coprocessor.
  - Symmetric Key Decipher (CSNBSYD) has been enhanced to support the DES and TDES algorithms, but requires CP Assist for Cryptographic Functions. *Rule_array* key processing rules CUSP, IPS, and X9.23 have been added.
  - Symmetric Key Encipher (CSNBSYE) has been enhanced to support the DES and TDES algorithms, but requires CP Assist for Cryptographic Functions. *Rule_array* key processing rules CUSP, IPS, and X9.23 has been added.
- Additional bit definitions (Crypto assist instructions and DES and TDES enablement) have been added to the Cryptographic Communication Vector Table (CCVT).

Deleted information

References to DATAC have been removed. The services affected are CV Generate, Key Export, Key Import, Key Generate, and Key Token Build. Double-length DATA keys should be used instead of DATAC.

References to Cryptographic Unit Support Product (CUSP) have been removed as the product is no longer supported.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of changes
for SA22-7520-03
z/OS Version 1 Release 3

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-02, which supports z/OS Version 1 Release 2.

New information

- Access Control Points
  - UKPT - PIN Verify, PIN Translate
- Callable services - The following new callable services perform encryption using the AES algorithm. AES encryption is only allowed if the CCC is enabled for triple DES. Only clear key support is provided.
  - Symmetric Key Decipher (CSNBSYD) - Deciphers data in an address space using the cipher block chaining or electronic code book modes.
  - Symmetric Key Encipher (CSNBSYE) - Enciphers data in an address space using the cipher block chaining or electronic code book modes.
- ICSF Setup
  - ICSF setup for E-Delivery delivery has been added. A sample ICSF options dataset, CSFPRM01, has been added to SYS1.SAMPLIB for the purpose of setting master keys by means of batch processing.
  - A sample CKDS allocation job (member CSFCKDS) has been added to SYS1.SAMPLIB.
  - A sample PKDS allocation job (member CSFPKDS) has been added to SYS1.SAMPLIB.
  - Samples for CSFSTART (ICSF Startup Procedures) has been added.
  - Sample JCL (CSFSETMK) for E-Delivery default passphrase has been added.
- Support to enable RMF to provide performance measurements on selected ICSF services and functions that use Direct Access Crypto (DAC) CCF instructions has been added.
- An appendix with z/OS product accessibility information has been added.

Changed information

- Callable services
  - Control Vector Generate (CSNBCVG) - *rule_array* enhanced to support the UKPT keyword.
  - Key Token Build (CSNBKTB) - *rule_array* enhanced to support the UKPT keyword.

  - Encrypted PIN Translate (CSNBPTR) - *rule_array* enhanced to support UKPT keywords UKPTIPIN, UKPTOPIN, and UKPTBOTH.
  - Encrypted PIN Verify (CSNBPVR) - *rule_array* enhanced to support UKPT keyword UKPTIPIN.
  - Symmetric Key Export (CSNDSYX) - a new *rule_array* keyword, PKCSOAEP, has been added. This keyword specifies the method found in RSA PKCS #1V2 OAEP.
  - Symmetric Key Generate (CSNDSYG) - a new *rule_array* keyword, PKCSOAEP, has been added. This keyword specifies the method found in RSA PKCS #1V2 OAEP.
  - Symmetric Key Import (CSNDSYI) - a new *rule_array* keyword, PKCSOAEP, has been added. This keyword specifies the method found in RSA PKCS #1V2 OAEP.
- The ICSF TSO panels have been updated to enhance usability:
  - Coprocessor management functions have been combined onto one panel
  - Master key management/CKDS functions combined onto one panel
  - TKE TSO utilities combined onto one panel
  - Primary panel simplified
  - New utility added to generate master key values from a pass phrase

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for SA22-7520-02
as Updated December 2001

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-01, which supports z/OS Version 1 Release 2.

Changed information

- Updated "Customize SYS1.PARMLIB" on page 10.
- Updated "Provide Access to the ICSF Panels" on page 18.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for SA22-7520-01
z/OS Version 1 Release 2

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-00, which supports z/OS Version 1 Release 1.

New information

- Callable services
  - PKA Key Token Change (CSNDKTC) callable service - This service changes PKA internal key tokens (RSA and DSS) from encipherment with the old PCI Cryptographic Coprocessor asymmetric-keys master key to encipherment with the current PCI Cryptographic Coprocessor asymmetric-keys master key.

  - Secure Messaging for Keys (CSNBSKY) callable service - This service encrypts a text block, including a clear key value decrypted from an internal or external DES token.
  - Secure Messaging for PINs (CSNBSPN) callable service - This service encrypts a text block, including a clear PIN block recovered from an encrypted PIN block.
- Installation Options Data Set
  - PKDSCACHE, an installation option, defines the size of the PKDS Cache in records. The PKDS cache improves performance as it facilitates access to frequently used records. Specify *n* as a decimal value from 0 to 256. If *n* is zero, no cache will be implemented. If PKDSCACHE is not specified, the default value is 64. PKDSCACHE can be implemented on OS/390 V2 R10 and z/OS V1 R1 by installing APAR OW48568.
  - When specifying parameter values within parentheses, leading and trailing blanks are ignored. Embedded blanks may cause unpredictable results.
- PCI Cryptographic Accelerator (PCICA) support has been added. If a PCI Cryptographic Accelerator is available, clear RSA key processing in the CSNDPKD service will be routed to the PCI Cryptographic Accelerator. If you have a PCI Cryptographic Accelerator online, toleration APAR OW49402 is required on lower levels of ICSF (OS/390 V2 R9, OS/390 V2 R10 and z/OS V1 R1).
- Support to REENCIPHER PKDS and ACTIVATE PKDS has been added to the Master Key Management Panels. The new utility, CSFPUTIL, can also be used to reencipher the PKDS from the old asymmetric-keys master key to the current master key and to activate the reenciphered PKDS. Toleration APAR OW49386 is required on the following systems in order to activate the re-enciphered PKDS:
  - HCRP210 (standalone), HCRP220(OS/390 V2 R6, OS/390 V2 R7, OS/390 V2 R8), HCRP230 (OS/390 V2 R9), and HCR7703 (OS/390 V2 R10 and z/OS V1 R1)
- UDX support - Support for writing your own UDX has been added.

Changed information

- Beginning in z/OS V1 R2, the DOMAIN parameter is an optional parameter in the installation options data set. It is, however, required if more than one domain is specified as the usage domain on the PR/SM panels or if running in native mode. If specified in the options data set, it will be used and it must be one of the usage domains for the LPAR. If DOMAIN is not specified in the options data set, ICSF determines which domains are available in this LPAR. If only one domain is defined for the LPAR, ICSF will use it. If more than one is available, ICSF will issue error message ″CSFM409E MULTIPLE DOMAINS AVAILABLE. SELECT ONE IN THE OPTIONS DATA SET.″
- Callable services
  - MAXLEN parameter checking has been eliminated for the following services:
    - Encipher (CSNBENC and CSNBENC1)
    - Decipher (CSNBDEC and CSNBDEC1)
    - MAC generate (CSNBMGN and CSNBMGN1)
    - MAC verify (CSNBMVR and CSNBMVR1)
    - Ciphertext translate (CSNBCTT and CSNBCTT1)
    - MDC generate (CSNBMDG and CSNBMDG1)

    The MAXLEN parameter is also no longer enforced in the CUSP compatibility CIPHER service. The MAXLEN parameter may still be specified in the options

data set, but only the maximum value limit will be enforced (2147483647). If a value greater than this is specified, an error will result and ICSF will not start.

- Pass Phrase Initialization now allows uninitialized PCI Cryptographic Coprocessors to be initialized without processing all Cryptographic Coprocessors. A new panel option (Initialize new PCICC Only) has been added to the Pass Phrase Initialization panel to allow the initialization of the new PCI Cryptographic Coprocessors.

Deleted information

- Message IEC161I has been eliminated during the first time startup of ICSF.
- The following reason codes for ICSF/MVS X'18F' are being eliminated and will be replaced with operator messages.
  - Reason Code X'3C' - replaced by message CSFM105E
  - Reason Code X'48' - replaced by message CSFM120E
  - Reason Code X'1B' - replaced by message CSFM410E
  - Reason Code X'4B' - replaced by message CSFM107E
  - Reason Code X'106' - If the CCC is all zeroes, abend X'18F' reason code 4A will occur. If the CCC does not exist, message CSFM113E will be displayed.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Chapter 1. Introduction to z/OS ICSF

ICSF is a software element of z/OS. ICSF works with the hardware cryptographic features and the Security Server (RACF element) to provide secure, high-speed cryptographic services in the z/OS environment. ICSF provides the application programming interfaces by which applications request the cryptographic services.ICSF is also the means by which the secure cryptographic features are loaded with master key values, allowing the hardware features to be used by applications. The cryptographic feature is secure, high-speed hardware that performs the actual cryptographic functions. Your processor hardware determines the cryptographic feature available to your applications.

Hardware Features

Cryptographic hardware features include cryptographic assist instructions, PCI Cryptographic Accelerator, PCI Cryptographic Coprocessor, and Cryptographic Coprocessor Feature.

Cryptographic Assist Instructions

The **IBM @server zSeries 990** provides constraint relief and addresses various customer demands. It has several cryptographic features.

- Cryptographic assist instructions are implemented on every processor. SHA-1 secure hashing is directly available to application programs.
- Feature code 3863, **CP Assist for Cryptographic Functions**, enables clear key DES and TDES instructions on all CPs. In addition, ICSF supports software implementation of AES.
- Feature code 0862, **PCI Cryptographic Accelerator**, is also available on the IBM @server zSeries 990. If you have a PCI Cryptographic Accelerator on your system, you must also have CP Assist Enablement. The IBM @server zSeries 990 can support a maximum of 12 PCI Cryptographic Accelerators.

Restriction: The IBM @server zSeries 990 does not support the Cryptographic Coprocessor Feature or PCI Cryptographic Coprocessor.

Cryptographic Coprocessor Feature

The **Cryptographic Coprocessor Feature** (CCF) can have up to two cryptographic coprocessors protected by tamper-detection circuitry and a cryptographic battery unit. The Cryptographic Coprocessor Feature is available on the following servers:

- IBM S/390 G6 Enterprise Server with feature code 0800 and one of the following feature codes: 0814, 0815, 0834, 0835
- IBM @server zSeries 800 with feature code 0800 plus feature code 0875.
- IBM @server zSeries 900 with feature code 0800 plus feature code 0875

Restriction: The IBM @server zSeries 990 does not support the Cryptographic Coprocessor Feature.

PCI Cryptographic Coprocessor

The **PCI Cryptographic Coprocessor** (PCICC) is based on the 4758 model 2 standard PCI-bus card package and is available on the following servers. You must have at least one Cryptographic Coprocessor Feature on your system with a PCICC.

- S/390 G6 Enterprise Server with feature codes 0864 and 0865. Feature code 0860 is needed for each PCI Cryptographic Coprocessor. Note that each feature code has one coprocessor.
- IBM @server zSeries 800 with feature codes 0861 and 0865. Note that each feature code has two coprocessors.
- IBM @server zSeries 900 with feature codes 0861 and 0865. Note that each feature code has two coprocessors.

Restriction: The IBM @server zSeries 990 does not support the PCI Cryptographic Coprocessor.

PCI Cryptographic Accelerator

The **PCI Cryptographic Accelerator** (PCICA) is available on the following servers. You must have at least one Cryptographic Coprocessor Feature on your system with a PCICA. Note that each feature code has two coprocessors.

- IBM @server zSeries 800 with feature code 0862.
- IBM @server zSeries 900 with feature code 0862.

Note: The IBM @server zSeries 800 and IBM @server zSeries 900 can support a combination of PCI Cryptographic Coprocessors (maximum of 16) or PCI Cryptographic Accelerators (maximum of 12), but the total must not exceed 16.

ICSF Features

ICSF protects data from unauthorized disclosure or modification. It protects data that is stored within a system, stored in a file on magnetic tape off a system, and sent between systems. It can also be used to authenticate identities of senders and receivers and to ensure the integrity of messages transmitted over a network. It uses cryptography to accomplish these functions.

Cryptography enciphers data, using an algorithm and a cryptographic key, so the data is in an unintelligible form. Deciphering data involves reproducing the intelligible data from the unintelligible data. To encipher and decipher data, ICSF uses either the U.S. National Institute of Science and Technology Data Encryption Standard (DES) algorithm, or the Commercial Data Masking Facility (CDMF).

The CDMF defines a scrambling technique for data confidentiality. The CDMF is intended to be a substitute for DES for those customers who have been previously prohibited from receiving IBM products that support DES data confidentiality services. The CDMF data confidentiality algorithm involves two processes: a key shortening process and a standard DES decryption and encryption process. A CDMF system uses an effectively shortened data key in the standard DES algorithm to encrypt and decrypt data. DES and CDMF are symmetric key algorithms, requiring the exchange of a secret key.

ICSF also supports several Public Key Algorithms (PKA), which do not require exchanging a secret key. You can use these algorithms to exchange DES or CDMF secret keys securely and to compute digital signatures for authenticating messages and users. For digital signatures, you use a pair of keys: a private (secret) key to sign a message and a corresponding public key to verify the signature. ICSF supports the RSA and DSS algorithms. (Refer to the Federal Information Processing Standard (FIPS) Publication 186 for DSS standards.)

A key can be any combination of hexadecimal characters. A key determines how ICSF uses the algorithm to uniquely encipher data.

You can call an ICSF callable service from an application program to perform a cryptographic function. ICSF uses keys in cryptographic functions to:

- Protect data
- Protect other keys
- Verify that messages were not altered between sender and receiver
- Generate, protect, and verify personal identification numbers (PINs)
- Distribute DES and CDMF keys
- Generate and verify digital signatures

You use ICSF callable services and programs to generate, maintain, and manage keys that are used in the cryptographic functions. A unique key performs each type of cryptographic function on ICSF. All DES keys, except the DES master key, are enciphered under another key. The DES master key, which is physically secure, enciphers each DES key that is used on the system. The symmetric-keys master key (SYM-MK) is a double-length key that is used only to encrypt other DES keys on the PCI Cryptographic Coprocessor. The SYM-MK must be the same as the DES master key on the Cryptographic Coprocessor Feature.

PKA master keys protect PKA private keys. There are two PKA master keys on the Cryptographic Coprocessor Feature. One PKA master key, the signature master key (SMK), protects private keys that are intended for creating digital signatures. The other PKA master key, the key management master key (KMMK), protects private keys that are used in DES key distribution. Private keys that are protected by the KMMK can also be used to generate digital signatures. The asymmetric-keys master key (ASYM-MK) on the PCI Cryptographic Coprocessor is a triple-length key used to encipher and decipher PKA keys. In order for the PCI Cryptographic Coprocessor to function, the hash pattern of the ASYM-MK must have the same value as the hash pattern of the SMK on the Cryptographic Coprocessor Feature. If the PCI Cryptographic Coprocessor master key values are different, then the PCI Cryptographic Coprocessor will not be made active.

Resource Access Control Facility (RACF), an element of the z/OS Security Server can be used to control access to cryptographic keys and functions.

The Cryptographic Key Data Set (CKDS)

Keys that are protected under the DES master key are stored in a VSAM data set that is called the cryptographic key data set (CKDS). ICSF provides a sample CKDS allocation job (member CSFCKDS) in SYS1.SAMPLIB. The CKDS contains individual entries for each key that is added to it. You can store all types of keys (except master keys and PKA keys) in the CKDS. Each record in the data set contains the key value encrypted under the master key and other information about the key. ICSF maintains two copies of the CKDS: a disk copy and an in-storage copy.

Restriction: Running FMID HCR7708 on an IBM @server zSeries 990 does not support the CKDS.

Note: When a CKDS record is written which contains a key token with a control vector that is not supported by the Cryptographic Coprocessor Feature, a key type of CV will be placed into the CKDS record. During CKDS reencipher processing, for any key containing a control vector which is not supported by the Cryptographic Coprocessor Feature, a key token change

request will be sent to the PCI Cryptographic Coprocessor to reencipher the key. In a sysplex with a shared CKDS, the CKDS reencipher process must be invoked on a system which has a PCI Cryptographic Coprocessor installed.

Callable services use the in-storage copy of the CKDS to perform CKDS functions. For information on managing and sharing the CKDS in a sysplex environment, see *z/OS ICSF Administrator's Guide*, SA22-7521. The key generator utility program (KGUP) updates the disk copy rather than the in-storage copy. Therefore, cryptographic functions do not have to stop while KGUP updates the CKDS. The ICSF administrator can use the ICSF panels or a utility program to refresh the in-storage CKDS with the updated disk copy of the CKDS. Applications can also use the dynamic CKDS update callable services to update both the in-storage and DASD copies of the CKDS with no interruption of cryptographic function.

To add operational keys to the CKDS, for S/390 Enterprise Servers, the S/390 Multiprise, the IBM @server zSeries 800, and the IBM @server zSeries 900, you can do the following:

- Use KGUP to generate or enter keys
- Use the dynamic CKDS update callable services to create and write keys directly to the CKDS
- For S/390 Enterprise Servers, the S/390 Multiprise, the IBM @server zSeries 800 and the IBM @server zSeries 900, you can use the Trusted Key Entry (TKE) workstation to load operational PIN and TRANSPORT keys. (TKE is not part of the base product. It is an optional feature.)

Note: See Appendix E, "Running HCR7708 on a IBM @server zSeries 990", on page 195 if you're running in this environment.

The Public Key Data Set (PKDS)

RSA and DSS public and private keys can be stored in a VSAM data set that is called the public key data set (PKDS). ICSF maintains the PKDS as an external data set. ICSF provides a sample PKDS allocation job (member CSFPKDS) in SYS1.SAMPLIB. In addition, ICSF optionally maintains a cache of frequently used PKDS records. The size of the PKDS cache is set in the installation options data set (PKDSCACHE). It is an optional feature, with a default of 64 records. If a cache is being maintained, care must be taken when deleting or changing an existing PKDS record. PKDSCACHE can be implemented on OS/390 V2R10 and z/OS V1 R1 by installing APAR OW48568.

Restriction: Running FMID HCR7708 on an IBM @server zSeries 990 does not support the PKDS.

You can store public key tokens or both external and internal private key tokens. Applications can use the dynamic PKDS update callable services to create, write, read, and delete PKDS records.

The PKDS is required for OS/390 V2 R9 ICSF and above, unless you are running FMID HCR7708 on an IBM @server zSeries 990.

Beginning in z/OS V1 R2, support to REENCIPHER PKDS and ACTIVATE PKDS has been added to the Master Key Management Panels and to the CSFPUTIL utility to reencipher the PKDS from the old ASYM-MK to the current master key and to activate the reenciphered PKDS. CSFPUTIL is a utility that performs the same functions as REENCIPHER PKDS and ACTIVATE PKDS. Other systems with lower

levels of ICSF which are sharing the PKDS would disable PKDS read and PKDS write, change the appropriate PKA master key(s),activate the reenciphered PKDS and enable PKDS Read and PKDS Write. For information on managing and sharing the PKDS in a sysplex environment, see *z/OS ICSF Administrator's Guide*, SA22-7521. Toleration APAR OW49386 is required on the following systems in order to activate the re-enciphered PKDS:

- HCRP210 (standalone), HCRP220(OS/390 V2 R6, OS/390 V2 R7, OS/390 V2 R8), HCRP230 (OS/390 V2 R9), and HCR7703 (OS/390 V2 R10 and z/OS V1 R1)

Note: See Appendix E, "Running HCR7708 on a IBM @server zSeries 990", on page 195 if you're running in this environment.

Additional Background Information

The following sections provide some additional background information about using ICSF with other products, such as the Programmed Cryptographic Facility (PCF).

Running PCF Applications on OS/390 ICSF

If your installation uses PCF, you can run PCF applications on ICSF. You can use an installation option to specify whether a PCF application runs on ICSF. If you are migrating from PCF, ICSF provides a conversion program that converts a PCF CKDS to ICSF format.

You can use your own installation services and exits to customize ICSF. You can write, define, and call your own installation-defined callable service. You can also write and define exits that ICSF calls during the processing of the following:

- ICSF mainline
- A callable service
- The PCF CKDS conversion program
- The key generator utility program
- CKDS access

For example, each callable service in ICSF calls an exit before and after processing. Such an exit can alter return codes in a service.

Running 4753-HSP Applications on ICSF

If your installation uses the IBM Network Security Processor Support Program products, you may be able to run 4753-HSP applications on ICSF without change. There are some restrictions when running both ICSF and 4753-HSP in the same z/OS (MVS) environment:

- Although both ICSF and 4753-HSP can run PCF compatibility mode, only one system can provide this service at any time.
- Because both ICSF and 4753-HSP support the Common Cryptographic Architecture (CCA) Application Programming Interface, applications need to be linked with the callable service stubs that each product provides to access the intended service.
- Internal key tokens are not interchangeable between the two products.

For more information on running ICSF and 4753-HSP in the same operating system environment, refer to "Running 4753-HSP and ICSF on the Same z/OS System" on page 71.

Using RMF and SMF to Monitor z/OS ICSF Events

You can run ICSF in different configurations and use installation options to affect ICSF performance. While ICSF is running, you can use the Resource Management Facilities (RMF) and System Management Facilities (SMF) to monitor certain events. For example, ICSF records information in the ICSF SMF data set when ICSF status changes in a processor or when you enter or change the master key. ICSF also sends information and diagnostic messages to data sets and consoles.

With the availability of cryptographic hardware (PCI Cryptographic Coprocessor and PCI Cryptographic Accelerator) on an LPAR basis, RMF provides performance monitoring in the Postprocessor Crypto Hardware Activity report. This report is based on SMF record type 70, subtype 2. The Monitor I gathering options on the REPORTS control statement are CRYPTO and NOCRYPTO. Specify CRYPTO to measure cryptographic hardware activity and NOCRYPTO to supress the gathering. In addition, overview criteria is shown for the Postprocessor in the Postprocessor Workload Activity Report - Goal Mode (WLMGL) report. Refer to *z/OS RMF Programmer's Guide*, SC33-7994, *z/OS RMF User's Guide*, SC33-7990, and *z/OS RMF Report Analysis*, SC33-7991 for additional information.

ICSF also supports enabling RMF to provide performance measurements on ICSF services (Encipher, Decipher, MAC Generate, MAC Verify, One Way Hash, PIN Translate, and PIN Verify) and functions that use Direct Access Crypto (DAC) CCF instructions.

For diagnosis monitoring, use Interactive Problem Control System (IPCS) to access the trace buffer and to format control blocks.

Controlling Access to ICSF

For security, you should control access to ICSF resources and services. Use a security product like the Security Server (RACF) to protect cryptographic programs, keys, and services. You should also change the value of the DES master key periodically. With PKDS reencipher, you can also change the PKA Master Key periodically.

Before Starting Installation

You use either ServerPac or CBPDO to install ICSF as part of the z/OS installation process.

Before beginning installation, do the following:

1. Refer to *z/OS and z/OS.e Planning for Installation* for installation planning information.
2. Check with your IBM center or search the IBM problem database to find any pertinent Preventative Service Planning (PSP). There may also be HOLDDATA and PSP information for ICSF on the tape.
3. Make sure that you have all needed programs and their corequisites:
   - If you use the Security Sever (RACF) and want access control and auditing services for ICSF, you need the Security Server (RACF), an optional feature of z/OS.
   - If you are a Resource Measurement Facility (RMF) user, you need the Resource Measurement Facility option available with z/OS.
4. Collect all required documents and information. The Program Directory lists publications useful during installation.

5. Confirm you have adequate DASD storage and create SMP/E DDDEF entries for each data set. See the Program Directory for details.

   Note: ICSF, for compatibility reasons, uses storage below the line in CSA (Common Storage Area). Some of the ICSF interface modules are loaded when ICSF is started and others are loaded while ICSF is running on an as needed basis. ICSF's CSA usage of below the line storage may be up to 36K.

Chapter 2. Installation, Initialization, and Customization

For this chapter, you need to understand the following terms:

installation options
: You create an installation options data set that specifies these options. They become active when you start ICSF, customizing how ICSF runs on your system.

startup procedure
: You create an ICSF startup procedure. Along with other information, this specifies the name of the installation options data set.

SYS1.SAMPLIB
: Contains samples, including an installation options data set, a CKDS allocation job, a PKDS allocation job, a startup procedure, a CICS Wait List data set, and sample JCL for SMP/E Delivery to load keys by using a passphrase. You can update this code as necessary and generally store the updated code in SYS1.PARMLIB and SYS1.PROCLIB.

SYS1.PARMLIB
: Generally contains the installation options data set. The installation options data set can alternately be a member of a partitioned or sequential data set.

SYS1.PROCLIB
: Contains the startup procedure.

Steps in Installation and Initialization

Refer to the *z/OS Program Directory* for installation instructions. Several of the installation steps in the *z/OS Program Directory* refer you to this document for details. This document explains the following installation steps.

Note: You only need to perform the first five steps once. If you stop ICSF and want to perform a subsequent SMP/E electronic delivery (this is optional), you only need to start ICSF (step 6). See Appendix E, “Running HCR7708 on a IBM @server zSeries 990”, on page 195 if you’re running in this environment.

1. Customize SYS1.PARMLIB. “Customize SYS1.PARMLIB” on page 10 provides details.
2. Create the Cryptographic Key Data Set (CKDS). “Create the CKDS” on page 10 describes this. Create the Public Key Data Set (PKDS). “Create the PKDS” on page 12 describes this.
3. Create the installation options data set. “Create the Installation Options Data Set” on page 14 provides details.
4. Create the startup procedure. “Create the ICSF Startup Procedure” on page 16 provides details.
5. Provide access to the ICSF panels. “Provide Access to the ICSF Panels” on page 18 describes this.
6. Start ICSF for the first time. See “Start ICSF for the First Time” on page 19. Once ICSF has been started, Master Keys can be entered. See *z/OS ICSF Administrator’s Guide* for directions on entering Master Keys.
7. Enter Master Keys.

8. Only complete this step if ICSF is needed for SMP/E. Do not complete this step for other applications.

   Run the JCL to set the SMP/E pass phrase for SMP/E electronic delivery (optional). “MK Initialization for SMP/E Only” on page 21 describes this.

Other chapters in this document, and *z/OS ICSF Administrator's Guide* provide additional installation information.

For information on installing the CICS-ICSF Attachment Facility, refer to Appendix B, “Installing the CICS-ICSF Attachment Facility”, on page 181.

For additional information on ICSF first time startup, refer to “Checklist for First-Time Startup of ICSF” on page 187.

Customize SYS1.PARMLIB

The installation options data set you will create is generally stored in SYS1.PARMLIB. If your administrator does not have access to SYS1.PARMLIB, you need to use another data set instead.

Update the data set you are using as follows:

1. Add CEE.SCEERUN and CSF.SCSFMOD0 to the LNKLST concatenation This adds the ICSF library to the z/OS library search. The following is an example of an ICSF entry to the LNKLST concatenation.

   ```
   CSF.SCSFMOD0
   ```

2. APF authorize CSF.SCSFMOD0, if LNKAUTH=APFTAB. The following is an example of an ICSF entry for APF authorization.

   ```
   APF ADD DSNAME(CSF.SCSFMOD0)  VOLUME(******)
   ```

3. In the IKJTSOxx parameter, add CSFDAUTH as a value in the AUTHPGM parameter list and in the AUTHTSF parameter list. The following is an example of an ICSF entry in the IKJTSOxx member.

   ```
   AUTHPGM NAMES(        /* AUTHORIZED PROGRAMS             */  +
    ....
    ....
   CSFDAUTH              /* ICSF                            */  +
   CSFDPKDS              /* ICSF                            */  +

    ....

   AUTHTSF NAMES(        /* PROGRAMS TO BE AUTHORIZED WHEN  */  +
                         /* WHEN CALLED THROUGH THE TSO     */  +
                         /* SERVICE FACILITY                */  +
    ....
    ....
   CSFDAUTH              /* ICSF                            */  +
   ```

4. If you will be using TKE V3.0 or later on this host, you should also add CSFTTKE as a value in the AUTHCMD parameter list.

   If you will only be using ICSF for SMP/E electronic delivery, this step does not need to be performed. TKE is not needed for SMP/E electronic delivery.

z/OS MVS Initialization and Tuning Guide and *z/OS MVS Initialization and Tuning Reference* provide more information.

Create the CKDS

The CKDS must be a key-sequenced data set with a fixed record length of 252 bytes. Allocate the CKDS on a permanently resident volume.

1. Determine the amount of primary space you need to allocate for the CKDS.

This should reflect the total number of entries you expect the data set to contain originally. Besides transport keys, PIN keys, data-encrypting keys, data-translating keys, and MAC keys, the CKDS contains a header record and system keys that ICSF uses for processing. To run ICSF requires the header record and four of the system keys. The other system keys, 48 NOCV enablement keys and 7 ANSI enablement keys, are optional.

Your system needs NOCV enablement keys if it communicates with systems that do not use control vectors, supports the use of DATA keys in the MAC services, or needs to convert a PCF CKDS. Your system needs ANSI enablement keys if you distribute keys according to the ANSI X9.17 protocol.

If you intend to use both NOCV enablement keys and ANSI enablement keys and you expect the CKDS to contain 100 transport, PIN, DATA, or MAC keys, allocate enough primary space for 100 records, 59 system key records, and a header record. For S/390 Enterprise Servers and S/390 Multiprise and IBM @server zSeries, there may also be three enhanced system keys.

To use the SET callable services on a CDMF system, you need to install both the NOCV keys and ANSI system keys.

2. Determine the amount of secondary space to allocate for CKDS.

   This should reflect the total number of entries you expect to add to the data set. For detailed information about calculating space for a VSAM data set, see *MVS/ESA VSAM Administration Guide*, SC26-4518.

   To access keys, VSAM uses the key label as the VSAM key. This means that VSAM adds keys to the data set in collating sequence. That is, if two keys named A and B are in the data set, A appears earlier in the data set than B. As a result, adding keys to the data set can cause multiple VSAM control interval splits and control area splits. For example, a split might occur if the data set contains keys A, B, and E and you add C. In this case, C must be placed between B and E. These splits can leave considerable free space in the data set and can affect KGUP performance.

   The amount of secondary space you allocate must take into account the number of control interval and control area splits that might occur. If the disk copy of the CKDS uses a significant amount of secondary space, you can copy it into another disk copy that you created with more primary space. You can do this by using the Access Method Services (AMS) REPRO command or the AMS EXPORT/IMPORT commands.

   The BUFFERSPACE parameter on the AMS DEFINE CLUSTER command (required by Step 3) lets VSAM optimize space for control area and control interval splits. For a detailed explanation of keyed-direct update processing and what happens when control area and control interval splits occur, see *MVS/DFP Managing VSAM Data Sets*.

3. Create an empty VSAM data set to use as the CKDS. ICSF provides a sample job to define the CKDS in member CSFCKDS of SYS1.SAMPLIB.

   Use the AMS DEFINE CLUSTER command to define the data set and to allocate its space.

   Note: To improve security and reliability of the data that is stored on the CKDS, do the following:

   - Use the ERASE and WRITECHECK parameters on the AMS DEFINE CLUSTER command. ERASE overwrites data records with binary zeros when the CKDS cluster is deleted. WRITECHECK provides hardware verification of all data that is written to the data set.
   - Create a Security Server (RACF) data set profile for the CKDS.

Allocate a disk copy of the CKDS by defining a VSAM cluster as in the following SYS1.SAMPLIB CSFCKDS member sample:

```
//CSFCKDS   JOB
//*******************************************************************
//* THIS JCL DEFINES A VSAM CKDS TO USE FOR ICSF                    *
//*                                                                 *
//*  CAUTION: This is neither a JCL procedure nor a complete JOB.   *
//*  Before using this JOB step, you will have to make the following *
//*  modifications:                                                 *
//*                                                                 *
//*  1) Add the job parameters to meet your system requirements.    *
//*  2) Be sure to change CSF to the appropriate HLQ if you choose  *
//*     not to use the default.                                     *
//*  3) The CKDS needs to be on a permanently resident volume.      *
//*                                                                 *
//*******************************************************************
//DEFINE  EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER (NAME(CSF.CSFCKDS)         -
                  VOLUMES(XXXXXX)           -
                  RECORDS(100 50)           -
                  RECORDSIZE(252,252)       -
                  KEYS(72 0)                -
                  FREESPACE(10,10)          -
                  SHAREOPTIONS(2)           -
                  UNIQUE)                   -
           DATA (NAME(CSF.CSFCKDS.DATA)     -
                  BUFFERSPACE(100000)       -
                  ERASE                     -
                  WRITECHECK)               -
          INDEX (NAME(CSF.CSFCKDS.INDEX))   -
/*
```

You can change and use the Job Control Language according to the needs of your installation. Please note that the JCL to define the CKDS differs from the JCL that defines the PKDS (RECORDSIZE and CISZ parameters). For more information about allocating a VSAM data set, see *MVS/ESA VSAM Administration Guide*, SC26-4518.

Create the PKDS

Starting with OS/390 V2 R9 ICSF, the PKDS must be allocated and specified on the PKDSN parameter of the options data set when you first start ICSF. ICSF support for the PCI Cryptographic Coprocessor requires a PKDS. Even if not available at first time start up, a PCI Cryptographic Coprocessor can be dynamically configured online on a S/390 G5 Enterprise Server or higher. Since ICSF can not tell if a PCI Cryptographic Coprocessor will be added, it requires the PKDS to be available at start up.

The PKDS must be a key-sequenced data set with variable length records. Allocate the PKDS on a permanently resident volume.

1. Determine the amount of primary space you need to allocate for the PKDS.

   This should reflect the total number of entries you expect the data set to contain originally. The PKDS will contain both public and private PKA keys. Each record has a maximum size of 2.8 KB. The average record length for a private key is 1 KB, and for a public key is 0.5 KB. Allocate space for a minimum of two private keys, one for digital signatures, and another for encipherment. In addition, allocate enough space for the number of public keys you expect to store in the PKDS. The number of public keys varies from system to system. Generally, only

those keys that are received from other users or systems are stored in the PKDS. The public keys are used to send messages to the owners of the public keys.

2. Determine the amount of secondary space to allocate for the PKDS.

   This should reflect the total number of entries you expect to add to the data set. For detailed information about calculating space for a VSAM data set, see *MVS/ESA VSAM Administration Guide*, SC26-4518.

   To access keys, VSAM uses the key label as the VSAM key. This means that VSAM adds keys to the data set in collating sequence. That is, if two keys named A and B are in the data set, A appears earlier in the data set than B. As a result, adding keys to the data set can cause multiple VSAM control interval splits and control area splits. For example, a split might occur if the data set contains keys A, B, and E and you add C. In this case, C must be placed between B and E.

   The amount of secondary space you allocate must take into account the number of control interval and control area splits that might occur. If the PKDS uses a significant amount of secondary space, you can copy it into another disk copy that you created with more primary space. You can do this by using the Access Method Services (AMS) REPRO command or the AMS EXPORT/IMPORT commands.

   The BUFFERSPACE parameter on the AMS DEFINE CLUSTER command (required by Step 3) lets VSAM optimize space for control area and control interval splits. For a detailed explanation of keyed-direct update processing and what happens when control area and control interval splits occur, see *MVS/DFP Managing VSAM Data Sets*.

3. Create an empty VSAM data set to use as the PKDS. Use the AMS DEFINE CLUSTER command to define the data set and to allocate its space. ICSF provides a sample job to define the PKDS in member CSFPKDS of SYS1.SAMPLIB.

   Note: To improve security and reliability of the data that is stored on the PKDS, do the following:

   - Use the ERASE and WRITECHECK parameters on the AMS DEFINE CLUSTER command. ERASE overwrites data records with binary zeros when the PKDS cluster is deleted. WRITECHECK provides hardware verification of all data that is written to the data set.
   - Create a Security Sever (RACF) data set profile for the PKDS.
   - The CISZ(8192) coded in the following sample in the DATA section is a hardcoded requirement.

4. Allocate a disk copy of the PKDS by defining a VSAM cluster as in the following SYS1.SAMPLIB CSFPKDS member sample:

```
//CSFPKDS   JOB
//*********************************************************************
//* THIS JCL DEFINES A VSAM PKDS TO USE FOR ICSF - FOR SMP/E         *
//*                                                                  *
//*  CAUTION: This is neither a JCL procedure nor a complete JOB.    *
//*  Before using this JOB step, you will have to make the following *
//*  modifications:                                                  *
//*                                                                  *
//*  1) Add the job parameters to meet your system requirements.     *
//*  2) Be sure to change CSF to the appropriate HLQ if you choose   *
//*     not to use the default.                                      *
//*  3) The PKDS needs to be on a permanently resident volume.       *
//*                                                                  *
//*********************************************************************
//DEFINE  EXEC PGM=IDCAMS,REGION=4M
```

```
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER (NAME(CSF.CSFPKDS)         -
                  VOLUMES(XXXXXX)           -
                  RECORDS(100,50)           -
                  RECORDSIZE(350,2800)      -
                  KEYS(72 0)                -
                  FREESPACE(0,0)            -
                  SHAREOPTIONS(2,3)         -
                  UNIQUE)                   -
           DATA (NAME(CSF.CSFPKDS.DATA)     -
                  BUFFERSPACE(100000)       -
                  ERASE                     -
                  CISZ(8192)                -
                  WRITECHECK)               -
         INDEX (NAME(CSF.CSFPKDS.INDEX))    -

/*
```

You can change and use the Job Control Language according to the needs of your installation. Please note that the JCL to define the PKDS differs from the JCL that defines the CKDS (RECORDSIZE and CISZ parameters). For more information about allocating a VSAM data set, see *MVS/ESA VSAM Administration Guide*, SC26-4518.

Create the Installation Options Data Set

The *installation options data set* is a file that you create that contains installation options. It becomes active when you start ICSF.

- The installation options data set can be a member of PARMLIB, a partitioned data set, a member of a partitioned data set, or a sequential data set.
- The format of each record in the data set must be fixed length or fixed block length.
- A physical line in the data set is 80 characters long. The system ignores any characters in positions 72 to 80 of the line.
- A logical line is one or more physical lines. You can group physical lines into a logical line by placing a comma at the end of the information. Only a comment can appear after the comma. The system ignores any other information between the comma and column 71.
- Continuation causes the next physical line to append immediately following the comma. The system removes all leading blanks on the next physical line.
- You can delimit comments by /* and */ and include them anywhere within the text. A comment cannot span physical records. The system removes comments from a logical line before parsing it. It ignores physical lines that contain only comments.
- Specify only one option setting or keyword on a logical line. (If you specify more than one, the system ignores all but the last one on the line. The system reports syntax errors, but the errors do not cause it to stop interpreting the file.)

ICSF provides two sample installation options data sets. These sample data sets use the recommended values for each option.

1. When you are starting ICSF for the first time:
   a. Change the name of the data set on the CKDSN and PKDSN statements to the name of the empty VSAM datasets you created earlier (in Step 3 on page 11 and Step 4 on page 13).
   b. Change SSM(NO) to SSM(YES).

c. For a complete description of options you may want to change after the first start, see “Customizing ICSF after the First Start” on page 22.)

2. Store the updated data set in SYS1.PARMLIB.

   Note: For convenience, the installation options data set generally resides in SYS1.PARMLIB. If your cryptographic administrator does not have update access to SYS1.PARMLIB, store installation options in another data set, and RACF-protect it.

The two sample installation options data sets are as follows in SYS1.SAMPLIB:

- CSFPRM00

```
/*                                          */
/*     LICENSED MATERIALS - PROPERTY OF IBM    */
/*                                          */
/*     "RESTRICTED MATERIALS OF IBM"           */
/*     5694-A01                             */
/*                                          */
/*     (C) COPYRIGHT IBM CORP. 1990, 2002      */
/*                                          */
/*                                          */
CKDSN(CSF.SCSFCKDS)
PKDSN(CSF.SCSFPKDS)
COMPAT(NO)
SSM(NO)
KEYAUTH(NO)
CHECKAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(64)
```

- CSFPRM01 (batch setup for SMP/E)

```
/*                                          */
/*     LICENSED MATERIALS - PROPERTY OF IBM    */
/*                                          */
/*     "RESTRICTED MATERIALS OF IBM"           */
/*     5694-A01                             */
/*                                          */
/*     (C) COPYRIGHT IBM CORP. 1990, 2002      */
/*                                          */
/*
CKDSN(CSF.SCSFCKDS)
PKDSN(CSF.SCSFPKDS)
COMPAT(NO)
SSM(YES)
KEYAUTH(NO)
CHECKAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(64)
```

Note: See “Changing Parameters in the Installation Options Data Set” on page 22 for descriptions of these parameters.

Starting with OS/390 V2 R9 ICSF, the use of system symbols in the options data set will be supported. System symbols can be used as values for any of the parameters. System symbols must be no more than 8 characters. ICSF allows the

CKDS and PKDS data set names to be a maximum of 44 characters with up to 21 qualifiers. See “Changing Parameters in the Installation Options Data Set” on page 22 for additional information.

The following example shows how system symbols could be used for the CKDS and PKDS data set names. You could use a SYS1.PARMLIB(IEASYMxx) file and modify CSFPRM01.

IEASYMxx file could contain:

```
/*-----------------------------------*/
/* SYSTEM SYMBOLS FOR ICSF CRYPTO    */
/*-----------------------------------*/
SYSDEF
 SYMDEF(&CKDSN001='CSF')
 SYMDEF(&CKDSN002='CSFCKDS')
 SYMDEF(&PKDSN001='CSF')
 SYMDEF(&PKDSN002='CSFPKDS')
```

CSFPRM01 could be modified as follows:

```
/*                                          */
/*     LICENSED MATERIALS - PROPERTY OF IBM    */
/*                                          */
/*     "RESTRICTED MATERIALS OF IBM"        */
/*     5694-A01                             */
/*                                          */
/*     (C) COPYRIGHT IBM CORP. 1990, 2002   */
/*                                          */
/*                                          */
CKDSN(&CKDSN001..&CKDSN002)
PKDSN(&PKDSN001..&PKDSN002)
COMPAT(NO)
SSM(YES)
KEYAUTH(NO)
CHECKAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(64)
```

When the machine or partition is IPLed, specify within the load parameter the symbol file that should be used. For example, if the above symbol file was called IEASYM01, then within the load member, the IEASYM entry might look like `IEASYM(00,01)`; where 00 denotes the IEASYM00 file (usually the system default) and 01 denotes the IEASYM01 file.

For SMP/E, CSFPRM01 can be copied to the CPAC.PARMLIB data set. The CKDS and PKDS data set names in CSFPRM01 need to match in Server-Pac. Outside of Server-Pac, you need to copy and edit CSFPRM01.

Create the ICSF Startup Procedure

ICSF provides the following two job control language programs. You can use this code as the basis for your startup procedure.

- member CSF in SYS1.SAMPLIB

  ```
  //CSF PROC
  //CSF EXEC PGM=CSFMMAIN,REGION=6M,TIME=1440
  //CSFLIST DD SYSOUT=A,LRECL=132,BLKSIZE=132,HOLD=YES
  //CSFPARM DD DSN=SYS1.PARMLIB(CSFPRM00),DISP=SHR
  ```

- member CSFSTART in SYS1.SAMPLIB (batch setup for SMP/E)

```
//CSFSTART PROC
//CSFSTART EXEC PGM=CSFMMAIN,REGION=6M,TIME=1440
//CSFLIST DD SYSOUT=A,LRECL=132,BLKSIZE=132,HOLD=YES
//CSFPARM DD DSN=SYS1.PARMLIB(CSFPRM01),DISP=SHR
```

Store this startup PROC in SYS1.PROCLIB (or another suitable library). For SMP/E this proc can be copied to the CPAC.PROC data set, and the data set name (SYS1.PARMLIB) can be copied to match the name you chose for the CPAC.PARMLIB data set (in which CSFPRM01 would be placed). This would work for Server-Pac. Outside of Server-Pac, you need to copy and edit CSFSTART.

1. Change or use the sample startup procedure according to your needs.
   a. In the sample code, the first line is the PROC statement. You can add one or more procedure variables to the PROC statement. For example, you can allow the system operator to decide at start time which member of the installation options data set to use. The following example allows the operator to enter START CSF,M=CSFPRM01, specifying an alternate set of start-up options.

      ```
      //CSF  PROC M=CSFPRM00
      :
      //CSFPARM DD DSN=MY.ICSF.PARM(&M),DISP=SHR;
      ```

      You can use the same principle to change the name of a sequential data set, if you are not using a partitioned data set.
   b. The third line is the CSFLIST DD statement. This must specify the CSFLIST data set. (The CSFLIST data set contains messages pertaining to running ICSF, input and output errors, and processing parameters. For an explanation of these messages, see *z/OS ICSF Messages*.) You can specify SYSOUT to direct the CSFLIST data set to a data set with suitable space and DCB characteristics.
   c. Also on the CSFLIST DD statement, you can add FREE=CLOSE. This is useful if a problem occurs during startup. When message CSFM001I is displayed, you can browse the CSFLIST data set or use a product to browse the SYSOUT file on the JES2 Spool data set.
   d. The last line is the CSFPARM DD statement. The sample code specifies SYS1.PARMLIB as the data set where the installation options data set is stored. If you stored the installation options data set elsewhere, replace SYS1.PARMLIB with the name of the data set where you stored the installation options.
   e. The CSFPARM DD statement also specifies member CSFPRM00 as the name of the installation options data set. If you used a different name when you created the installation options data set (or any time you want to use other options), change this member name.
2. Store your startup procedure in SYS1.PROCLIB (or another suitable library) with a member name of your choice. (Depending on installation standards, possible names include CSF, CSFPROD, CSFTEST, and CRYPTO.)
3. If you use Security Server (RACF), you may need to update the RACF Started Procedure Table if you define a new started task:
   a. Add the new started task name
   b. Add a RACF userid to associate with the started task. This userid requires the following:
      - READ access to the data set to which the CSFPARM JCL DD statement refers

- If it refers to a data set, UPDATE access to the CSFLIST statement JCL DD statement (SYSOUT requires no RACF authority)
- Define all CKDSs in every installation option data set.

c. Optionally, you can add a RACF group name.

Note: RACF uses the userid associated with the ICSF address space only when accessing the CKDS named in the installation options data set and then only at ICSF startup. When you perform a CKDS Refresh task by using the ICSF ISPF panels under TSO/E, RACF uses the TSO userid to determine access authorization. When the CKDS Refresh task is a batch job, RACF uses the userid associated with the batch address space to determine access authorization.

Provide Access to the ICSF Panels

To provide a way for the administrator to access the ICSF panels, you can create an ICSF option on the ISPF Primary Option Menu. Access the code for the ISPF Primary Option Menu panel body and perform the following steps:

1. Under the % OPTION ===> _ZCMD line, add the following line:

   ```
   % <option value> - ICSF Panels
   ```

   You can specify either a letter or number for the option value. Do not use an option value that already exists in the menu.
2. On the &ZSEL= TRANS(&ZQ line, add the following information:

   ```
   <option value>,'PANEL(CSF@PRIM)'
   ```

   The option value should be the same value as the option value you chose to use in the preceding step.

When you access the ISPF Primary Option Menu panel, the ICSF panels option appears on the menu. You can choose the ICSF option value to access the ICSF panels.

You must also update the logon procedure that is used by ICSF administrators who will use the ICSF panels. For example:

```
//SYSPROC DD ...
.
.
.
//        DD  DSN=CSF.SCSFCLI0,DISP=SHR
.
.
.
//ISPPLIB DD ...
.
.
.
//        DD  DSN=CSF.SCSFPNL0,DISP=SHR
.
.
.
//ISPMLIB DD ...
.
.
.
//        DD  DSN=CSF.SCSFMSG0,DISP=SHR
.
.
.
```

```
//ISPSLIB DD ...
.
.
.
//        DD  DSN=CSF.SCSFSKL0,DISP=SHR
.
.
.
// ISPTLIB
.
.
.
//        DD DSN=CSF.SCSFTLIB,DISP=SHR
.
.
.
```

An alternate method to access the ICSF panels is to use ISPF LIBDEF. Here is a sample clist.

```
/* Rexx */
/* IBMs ICSF */

 address ispexec

"LIBDEF ISPPLIB DATASET ID('CSF.SCSFPNL0') STACK"
"LIBDEF ISPMLIB DATASET ID('CSF.SCSFMSG0') STACK"
"LIBDEF ISPSLIB DATASET ID('CSF.SCSFSKL0') STACK"
"LIBDEF ISPTLIB DATASET ID('CSF.SCSFTLIB') STACK"

address tso "ALTLIB ACTIVATE APPLICATION(CLIST)
            DATASET('CSF.SCSFCLI0')"
"SELECT PANEL(CSF@PRIM)"
address tso "ALTLIB DEACTIVATE APPLICATION(CLIST)"

"LIBDEF ISPSLIB"
"LIBDEF ISPPLIB"
"LIBDEF ISPMLIB"
"LIBDEF ISPTLIB"
```

Ensure that the latest CSFKEYS file is part of ISPTLIB; this allows scrolling of the management panels.

The *z/OS Program Directory* lists additional installation steps, and some of these steps depend on the system from which you are migrating. See *z/OS Program Directory*, other chapters in this document, and *z/OS ICSF Administrator's Guide* for details about the remaining steps.

Start ICSF for the First Time

If you are running on an IBM @server zSeries 990, see Appendix E, "Running HCR7708 on a IBM @server zSeries 990", on page 195.

Before starting ICSF for the first time, ensure that you have completed the following steps. For additional information on starting ICSF for the first time, see Appendix C, "Helpful Hints for ICSF First Time Startup", on page 187.

- Created an empty data set for use as a CKDS
- Specified the CKDS name in the installation options data set
- Created an empty data set for use as a PKDS (required for OS/390 V2 R9 ICSF or higher)
- Specified the PKDS name in the installation options data set (required for OS/390 V2 R9 ICSF or higher)
- Created a startup procedure

- Installed ICSF

Before using ICSF you must initialize it, which you are now ready to do.

1. Enter the START command and the startup procedure name. In the following example, CSF is the name of the startup procedure.

   ```
   START CSF
   ```

   When you start ICSF, you specify the name of the ICSF startup procedure you created (see "Create the ICSF Startup Procedure" on page 16). See "Starting and Stopping ICSF" on page 130 for more information about starting and stopping ICSF.

   Note: If you start CSF using CSFSTART and then run the CSFSETMK JCL to set the master keys and initialize the CKDS, the DES master keys will be set and the PKA master keys will be set in the Cryptographic Coprocessor Feature, and the CKDS will be initialized using the appropriate pass phrase. If your environment has PCI Cryptographic Coprocessors, they will not be initialized by this process. Only the Cryptographic Coprocessor Feature is initialized. If you need to initialize the PCI Cryptographic Coprocessor, see *z/OS ICSF Administrator's Guide* for additional information on using the Pass Phrase Initialization Utility. If you re-IPL or stop ICSF and want to perform a subsequent SMP/E E-delivery, you only need to start ICSF (providing you wish to reuse the previously established options and parameters).
2. Access the ICSF panels to define a master key and initialize the CKDS. For a description of how to use the ICSF panels to define a master key and initialize the CKDS at first-time startup, see *z/OS ICSF Administrator's Guide*.

Notes:

1. When you start ICSF for the first time, you'll see the following messages. You'll receive message CSFM511E for each Cryptographic Coprocessor Feature you have online.

   ```
   S CSFEC70
   $HASP100 CSFEC70  ON STCINRDR
   IEF695I START CSFEC70  WITH JOBNAME CSFEC70  IS ASSIGNED TO USER
   ++++++++
   $HASP373 CSFEC70  STARTED
   IEF403I CSFEC70 - STARTED - TIME=10.11.35
   CSFM100E CRYPTOGRAPHIC KEY DATA SET, ECHAN.DOMAIN7.CKDS IS NOT
   INITIALIZED.
   CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR 0, CPU 0 IS NOT VALID.
   CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR 1, CPU 4 IS NOT VALID.
   CSFM106A CRYPTOGRAPHY - PKA MASTER KEYS ARE NOT VALID.
   CSFM411I PCI CRYPTOGRAPHIC ACCELERATOR Axx IS ACTIVE.
   CSFM001I ICSF INITIALIZATION COMPLETE
   ```
2. When you are starting ICSF for the first time and loading the first master key and initializing one or more CKDSs, you provide the name of the empty VSAM data set you defined earlier (see step 3 on page 11) to use for the CKDS when starting ICSF.
3. While ICSF processes the data set, it requires exclusive use so that no one can make changes while the data set is read. ICSF releases the data set when it completes startup processing.
4. During CKDS initialization or refresh, ICSF reads the CKDS into extended private storage. Make sure that the region size is sufficient for reading in the entire data set.

5. You can add keys to the CKDS in several ways. See “The Cryptographic Key Data Set (CKDS)” on page 3 for details.
6. You can also write application programs to call services to perform cryptographic functions. See “Creating ICSF Exits and Generic Services” on page 31 for details.

MK Initialization for SMP/E Only

This is not necessary if you are running FMID HCR7708 on an IBM @server zSeries 990. See Appendix E, “Running HCR7708 on a IBM @server zSeries 990”, on page 195.

Run the JCL to set the SMP/E pass phrase for SMP/E electronic delivery only. The JCL uses a pass phrase value to load the DES and PKA master keys. The DES and PKA master keys will be set in the Cryptographic Coprocessor Feature. *Change This Pass Phrase* is the default pass phrase. The entry point is CSFEUTIL and will have 2 or (optionally) 3 parameters. The first parameter must be the CKDS name. The second parameter (optional) is the pass phrase. The last parameter is the function PPINIT. If you do not use the default pass phrase and create your own:

- It must be sixteen to sixty-four bytes in length.
- Any EBCDIC character is allowed.
- Leading and trailing blanks will be removed.
- Embedded blanks are allowed.

See the example below:

```
//CSFSETMK  JOB (JOB CARD PARAMETERS)
//*******************************************************************
//*  Licensed Materials - Property of IBM                           *
//*  5694-A01                                                       *
//*  (C) Copyright IBM Corp. 2002                                   *
//*                                                                 *
//* THIS JCL USES A PASS PHRASE VALUE TO LOAD DES AND PKA MASTER KEYS*
//*                                                                 *
//*  CAUTION: This is neither a JCL procedure nor a complete JOB.   *
//*  Before using this JOB step, you will have to make the following *
//*  modifications:                                                 *
//*                                                                 *
//*  1) Add the job parameters to meet your system requirements.    *
//*  2) The first parameter must be the CKDS name                   *
//*  3) An optional second parameter may be used.  The second       *
//*     parameter must be 16-64 character pass phrase.              *
//*     For the pass phrase any EBCDIC character is allowed.        *
//*     Leading and trailing blanks will be removed.                *
//*     Embedded blanks are allowed.                                *
//*     It is STRONGLY recommended that the pass phrase NOT contain *
//*     any commas.  Commas are used as a delimiter for the         *
//*     parameters of the CSFEUTIL program.                         *
//*  4) The last parameter must be the function PPINIT.             *
//*  5) If the default pass phrase of "Change This Pass Phrase"     *
//*     is desired, the PARM= would look like this:                 *
//*        PARM='CSF.CSFCKDS,PPINIT'                                *
//*                                                                 *
//*     If a customer selected pass phrase is to be used the        *
//*     PARM= would look like this:                                 *
//*        PARM='CSF.CSFCKDS,Different Pass Phrase,PPINIT'          *
//*                                                                 *
//*******************************************************************
//*     User supplied pass phrase of Different Pass Phrase
//STEP     EXEC PGM=CSFEUTIL,
//   PARM='CSF.CSFCKDS,Different Pass Phrase,PPINIT'
//SYSPRINT DD SYSOUT=*
```

```
//*

 OR:

//*    Using the default pass phrase of Change This Pass Phrase
//STEP     EXEC PGM=CSFEUTIL,
//   PARM='CSF.CSFCKDS,PPINIT'
//SYSPRINT DD SYSOUT=*
//*
```

In order to successfully run the CSFSETMK job, determine if the following services are RACF protected in the CSFSERV class. If the services below are not RACF protected in the CSFSERV class, then nothing needs to be done. If the services are protected in the CSFSERV class, then the issuer of the CSFSETMK JCL must be permitted to the profile for each service.

- CSFOWH
- CSFPMCI
- CSFCMK
- CSFREFR

Customizing ICSF after the First Start

See Appendix E, “Running HCR7708 on a IBM @server zSeries 990”, on page 195 if you’re running in this environment.

The startup procedure includes a CSFPARM DD statement, which gives the name of the installation options data set. The installation options data set includes a CKDSN option, which gives the names of the CKDS, and a PKDSN option, which gives the name of the PKDS.

After the first start, whenever you restart ICSF, the CKDS named in the installation options data set becomes the active in-storage CKDS.

To change the active in-storage CKDS (or any other installation option), change the option value in the installation options data set and **stop and restart ICSF**.

Changing Parameters in the Installation Options Data Set

The installation options data set is an intended programming interface.

When specifying parameter values within parentheses, leading and trailing blanks are ignored. Embedded blanks may cause unpredictable results.

Starting with OS/390 V2 R9 ICSF, support is provided for the use of system symbols in the installation options data set. System symbols can be used as values for any of the parameters. System symbols are specified in the IEASYMxx member of SYS1.PARMLIB; the IEASYM statement of the LOADxx member of SYS1.PARMLIB specifies the IEASYMxx member(s) to be used for the resolution of system symbols. The following example shows the use of a system symbol for specifying the domain to be used for the start of ICSF:

```
DOMAIN(&PARDOM.)
```

When the Installation Options Data Set is processed during the start of ICSF, the value of the system symbol PARDOM will be substituted as the value of the DOMAIN parameter.

For the first start, you specified an empty VSAM data set name for the CKDS in the CKDSN option, an empty VSAM data set name for the PKDS in the PKDSN option, and SSM(YES). You may want to change these and other options for subsequent starts. Here is a complete list of installation options:

CHECKAUTH(YES or NO)
Indicates whether ICSF performs security access control checking of Supervisor State and System Key callers. If you specify CHECKAUTH(YES), ICSF performs the checking. If you specify CHECKAUTH(NO), this results in significant performance enhancement for Supervisor State and System Key callers.

If you do not specify the CHECKAUTH option, the default is CHECKAUTH(NO).

CKDSN(data-set-name)
Specifies the CKDS name the system uses to start ICSF. Whenever you restart ICSF, the CKDS named in the CKDSN option becomes the active in-storage CKDS. (At first-time startup, you should specify the name of an empty VSAM data set you created to use as the CKDS.)

If you do not specify this keyword, ICSF does not become active. There is no default for this option, so you must specify a value.

COMPAT(YES, NO, or COEXIST)
Indicates whether ICSF runs in compatibility mode, non-compatibility mode, or coexistence mode with PCF.

YES Indicates **compatibility mode**.

In compatibility mode, you can run a PCF application on ICSF, because ICSF supports the PCF macros. You do not have to reassemble PCF applications to do this. You cannot start PCF at the same time as ICSF on the same operating system.

NO Indicates **non-compatibility mode**. In noncompatibility mode, you run PCF applications on PCF and ICSF applications on ICSF. You cannot run PCF applications on ICSF, because ICSF does not support the PCF macros in this mode. PCF can be started at the same time as ICSF on the same operating system. You can start ICSF and then start PCF, or you can start PCF and then start ICSF.

You should use noncompatibility mode unless you are migrating from PCF to ICSF.

COEXIST Indicates **coexistence mode**.

In coexistence mode, you can run a PCF application on PCF, or you can reassemble the PCF application to run on ICSF. To do this, you reassemble the application against coexistence macros that are shipped with ICSF. You can start PCF at the same time as ICSF on the same operating system.

If you do not specify the COMPAT option, the default value is COMPAT(NO). See “Running PCF and z/OS ICSF on the Same System” on page 33 for a complete description of the COMPAT options.

When you initialize ICSF for the first time, noncompatibility mode must be active. Therefore, at first-time startup, you must specify COMPAT(NO) or allow the default to be used.

COMPENC(DES or CDMF)

Specifies the encryption algorithm to use for the PCF compatibility CIPHER macro. For S/390 Enterprise Servers and S/390 Multiprise processors only, you can select the ANSI Data Encryption Standard (DES) or the Commercial Data Masking Facility (CDMF). For bipolar processors, only DES is available.

Your system can be DES-only, CDMF-only, or DES-CDMF. Specify COMPENC(DES) for a DES-only system. Specify COMPENC(CDMF) for a CDMF-only system. These are the defaults for these types of systems. If you specify an incorrect keyword (for example, DES for a CDMF-only system), the CIPHER macro fails with RC = 4 and the cryptographic facility is not available.

You can specify either COMPENC(DES) or COMPENC(CDMF) for a DES-CDMF system; the default is DES.

DOMAIN(n)

Specifies the number of the domain that you want to use for this start of ICSF. You can specify only one domain in the options data set.

Beginning with z/OS V1 R2, DOMAIN is an optional parameter. The DOMAIN parameter is only required if more than one domain is specified as the usage domain on the PR/SM panels or if running in native mode. If specified in the options data set, it will be used and it must be one of the usage domains for the LPAR.

If DOMAIN is not specified in the options data set, ICSF determines which domains are available in this LPAR. If only one domain is defined for the LPAR, ICSF will use it. If more than one is available, ICSF will issue error message CSFM409E.

The cryptographic processors support multiple sets of master key registers, which the specific domain values identify. The Cryptographic Coprocessor Feature has a master key register for the DES master key, the auxiliary DES master key, the signature master key and the key management master key. The auxiliary DES master key register may contain either the new or old DES master key. On the PCI Cryptographic Coprocessor, each domain has a master key register for the current, new, and old SYM-MK and ASYM-MK.

For more information about partitions and running different configurations, see “Using Different Configurations” on page 131.

If you run ICSF in compatibility or coexistence mode, you cannot change the domain number without re-IPLing the system. A re-IPL ensures that a program does not access a cryptographic service with a key that is encrypted under a different master key. If you are certain that no cryptographic applications are still running, you can do the following:

1. Stop CSF
2. Start CSF in COMPAT(NO) mode with a different domain number
3. Stop CSF
4. Start CSF in compatibility or coexistence mode with a different domain number.

EXIT(ICSF-name,load-module-name,FAIL(fail-option))

Indicates information about an installation exit.

The *ICSF-name* is the identifier for each exit. Table 1 lists all the ICSF exit names and explains when ICSF calls each exit. The load module name is the name of the module that contains the exit. The name can be any valid name your installation chooses.

Using the FAIL keyword of the EXIT statement, you specify the action ICSF, the KGUP, or the PCF conversion program takes if the exit ends abnormally. The fail action that you specify applies to subsequent calls of the exit. If an exit ends abnormally, ICSF takes a system dump. The exit is protected with an ESTAE or the ICSF service functional recovery routine (FRR).

In general, you can specify one of the following values for a fail option:

NONE No action is taken. The exit can be called again and will end abnormally again.

EXIT The exit is no longer available to be called again.

SERVICE
The service or program that called the exit is no longer available to be called again.

ICSF ICSF or the key generator utility program or the PCF conversion program is ended, depending on the exit.

Some fail options are not valid for a specific exit. If you specify a fail option that is not valid, ICSF uses the next valid fail option. For example, if SERVICE is not a valid fail option for an exit, ICSF uses the EXIT option.

Table 1. Exit Identifiers and Exit Invocations

Exit Identifiers	Exit Invocations
CSFEXIT1	Gets control after the operator issues the START command, but before processing takes place. **Note:** You must not specify an EXIT statement for the first mainline exit, CSFEXIT1.
CSFEXIT2	Gets control after ICSF reads and interprets the installation options data set.
CSFEXIT3	Gets control before ICSF completes initialization.
CSFEXIT4	Gets control after the operator issues the STOP command to stop ICSF.
CSFEXIT5	Gets control when the operator issues the MODIFY command to modify ICSF.
CSFEMK	Gets control during the compatibility service for the PCF EMK macro.
CSFGKC	Gets control during the compatibility service for the PCF GENKEY macro.
CSFRTC	Gets control during the compatibility service for the PCF RETKEY macro.
CSFEDC	Gets control during the compatibility service for the PCF CIPHER macro.
CSFCKDS	Gets control when a callable service retrieves an entry from the CKDS.
CSFKGUP	Gets control during key generator utility program initialization, processing, and termination.
CSFCONVX	Gets control when you run the PCF CKDS conversion program.
CSFSRRW	Gets control when an access to a single record in the CKDS is made using the key entry hardware.
CSFAEGN	Gets control during the ANSI X9.17 EDC generate callable service.
CSFAKEX	Gets control during the ANSI X9.17 key export callable service.

Table 1. Exit Identifiers and Exit Invocations (continued)

Exit Identifiers	Exit Invocations
CSFAKIM	Gets control during the ANSI X9.17 key import callable service.
CSFAKTR	Gets control during the ANSI X9.17 key translate callable service.
CSFAKTN	Gets control during the ANSI X9.17 transport key partial notarize callable service.
CSFCKI	Gets control during the clear key import callable service.
CSFCPE	Gets control during the clear PIN encrypt callable service.
CSFCPA	Gets control during the clear PIN generate alternate callable service.
CSFCTT	Gets control during the ciphertext translate callable service.
CSFCTT1	Gets control during the ciphertext translate (with ALET) callable service.
CSFPGN	Gets control during the Clear PIN generate callable service.
CSFCVT	Gets control during the control vector translate callable service.
CSFCVE	Gets control during the cryptographic variable encipher callable service.
CSFDKX	Gets control during the data key export callable service.
CSFDKM	Gets control during the data key import callable service.
CSFDEC	Gets control during the decipher callable service.
CSFDEC1	Gets control during the decipher (with ALET) callable service.
CSFDCO	Gets control during the decode callable service.
CSFDSG	Gets control during the digital signature generate service.
CSFDSV	Gets control during the digital signature verify callable service.
CSFDKG	Gets control during the diversified key generate callable service.
CSFENC	Gets control during the encipher callable service.
CSFENC1	Gets control during the encipher (with ALET) callable service.
CSFECO	Gets control during the encode callable service.
CSFEPG	Gets control during the encrypted PIN generate callable service.
CSFPTR	Gets control during the encrypted PIN translate callable service.
CSFPVR	Gets control during the encrypted PIN verify callable service.
CSFKEX	Gets control during the key export callable service.
CSFKGN	Gets control during the key generate callable service.
CSFKIM	Gets control during the key import callable service.
CSFKPI	Gets control during the key part import callable service.
CSFKRC	Gets control during the key record create callable service.
CSFKRD	Gets control during the key record delete callable service.
CSFKRR	Gets control during the key record read callable service.
CSFKRW	Gets control during the key record write callable service.
CSFKYT	Gets control during the key test callable service.
CSFKYTX	Gets control during the key test extended callable service.
CSFMDG	Gets control during the MDC generate callable service.
CSFKTR	Gets control during the key translate callable service.
CSFMGN1	Gets control during the MAC generate (with ALET) callable service.
CSFMVR	Gets control during the MAC verify callable service.
CSFMVR1	Gets control during the MAC verify (with ALET) callable service.

Table 1. Exit Identifiers and Exit Invocations (continued)

Exit Identifiers	Exit Invocations
CSFMDG1	Gets control during the MDC generate (with ALET) callable service.
CSFMGN	Gets control during the MAC generate callable service.
CSFCKM	Gets control during the multiple clear key import callable service.
CSFSKM	Gets control during the multiple secure key import callable service.
CSFOWH	Gets control during the one-way hash generate callable service.
CSFOWH1	Gets control during the one-way hash generate (with ALET) callable service.
CSFPCI	Gets control during the PCI interface callable service.
CSFPEX	Gets control during the prohibit export callable service.
CSFPEXX	Gets control during the prohibit export extended callable service.
CSFPKD	Gets control during the PKA decrypt callable service.
CSFPKE	Gets control during the PKA encrypt callable service.
CSFPKG	Gets control during the PKA key generate callable service.
CSFPKI	Gets control during the PKA key import callable service.
CSFPKTC	Gets control during the PKA key token change callable service.
CSFPKX	Gets control during the PKA Public Key Extract callable service.
CSFPKRC	Gets control during the PKDS record create callable service.
CSFPKRD	Gets control during the PKDS record delete callable service.
CSFPKRR	Gets control during the PKDS record read callable service.
CSFPKRW	Gets control during the PKDS record write callable service.
CSFPKSC	Gets control during the PKSC interface callable service.
CSFRNG	Gets control during the random number generate callable service.
CSFRKD	Gets control during the retained key delete callable service.
CSFRKL	Gets control during the retained key list callable service.
CSFSKI	Gets control during the secure key import callable service.
CSFSKY	Gets control during the secure messaging for keys callable service.
CSFSPN	Gets control during the secure messaging for PINs callable service.
CSFSBC	Gets control during the SET block compose callable service.
CSFSBD	Gets control during the SET block decompose callable service.
CSFSYX	Gets control during the symmetric key export callable service.
CSFSYG	Gets control during the symmetric key generate callable service.
CSFSYI	Gets control during the symmetric key import callable service.
CSFTCK	Gets control during the transform CDMF key callable service.
CSFUDK	Gets control during the user derived key callable service.
CSFCSG	Gets control during the VISA CVV service generate callable service.
CSFCSV	Gets control during the VISA CVV service verify callable service.

See Chapter 7, “Installation Exits”, on page 83 for a detailed description of each ICSF exit, including the valid fail options.

Note: z/OS no longer ships IBM-supplied security exit routines; the security exit points remain. Users of z/OS should use the Security Server

(RACF) or an equivalent product to obtain access checking of services and keys. ICSF no longer needs these exit routines.

KEYAUTH(YES or NO)

Indicates whether or not ICSF authenticates a key entry after ICSF retrieves one from the in-storage CKDS. If you specify KEYAUTH(YES), ICSF authenticates the key. ICSF generates a message authentication code (MAC) for each key entry in the CKDS when you create or update the entry. If you specify KEYAUTH(YES), ICSF performs a MAC verification to ensure that the entry has not changed. If you specify KEYAUTH(NO), ICSF does not perform this authentication and gains a small performance enhancement. If you do not specify the KEYAUTH option, the default value is KEYAUTH(NO).

MAXLEN(n)

Defines the maximum length of characters in a text string, including any necessary padding, for some callable service requests. For example, this option defines the maximum length of the text the encipher service encrypts for each call. Specify *n* as a decimal value from 1024 through 2147483647. If you do not specify the MAXLEN option, the default value is MAXLEN(65535).

Beginning with z/OS V1 R2, the MAXLEN parameter may still be specified in the options data set, but only the maximum value limit will be enforced (2147483647). If a value greater than this is specified, an error will result and ICSF will not start.

Note: Beginning with z/OS V1 R2, MAXLEN is no longer displayed on the Installation Option Display panel.

PKDSCACHE(n)

Defines the size of the PKDS Cache in records. The PKDS cache improves performance as it facilitates access to frequently used records. Specify *n* as a decimal value from 0 to 256. If *n* is zero, no cache will be implemented. If PKDSCACHE is not specified, the default value is 64. PKDSCACHE can be implemented on OS/390 V2R10 and z/OS V1 R1 by installing APAR OW48568.

PKDSN(data-set-name)

Specifies the PKDS name that ICSF initializes. With previous releases of OS/390, this keyword was optional. Starting with OS/390 V2 R9 ICSF, however, this keyword is now required. ICSF will not start without it. Specifying this keyword in your installation options data set allows you to store PKA key tokens in this VSAM data set. You can then refer to these tokens by key label in your applications.

There is no default for this option.

REASONCODES(ICSF or TSS)

Specifies which set of reason codes are to be returned from callable services. If you do not specify the REASONCODES option, the default of REASONCODES(ICSF) is used. If you specify REASONCODES(TSS), TSS reason codes will be returned. If there is a 1-to-1 mapping, the codes will be converted. If there is not a map to ICSF, the column will be blank. If there are multiple mappings, they will be listed as reference only and will not be converted.

SERVICE(service-number,load-module-name,FAIL(fail-option))

Indicates information about an installation-defined service.

ICSF allows an installation to define its own service similar to an ICSF callable service. The *service-number* specifies a number that identifies the service to ICSF. The valid service numbers are 1 through 32767, inclusive. This set of service numbers is valid for both installation-defined services and UDX services. The service number of an installation-defined service must not be the same as the service number of a UDX service. The *load-module-name* is the name of the module that contains the service. During ICSF startup, ICSF loads this module and binds it to the *service-number* you specified.

The *fail-option* is YES or NO, indicating the action ICSF should take if loading the service ends abnormally.

YES Specifies that ICSF ends abnormally if your service cannot be loaded.

NO Specifies that ICSF continues to start if your service cannot be loaded.

If the service itself ends abnormally, ICSF does not end, but takes a system dump instead. The ICSF service functional recovery routine (FRR) protects the service.

See Chapter 6, “Installation-Defined Callable Services”, on page 73 for a description of how to write and run an installation-defined callable service.

SSM(YES or NO)

Specifies whether or not an installation can enable special secure mode (SSM) while running ICSF. SSM lowers the security of your system to let you enter clear keys and generate clear PINs. You must enable SSM for KGUP to permit generation or entry of clear keys and to enable the secure key import or clear pin generate callable services.

YES Indicates that you can enable the SSM.

NO Indicates that you cannot enable the SSM.

If you do not specify the SSM option, the default value is SSM(NO).

Note: When you initialize ICSF for the first time, SSM must be active. Therefore, at first-time startup, you must specify SSM(YES).

You must perform the following tasks to make SSM active:
- Specify SSM(YES) in the installation options data set
- Enable SSM in the cryptographic hardware
- When running under a logical partition (LPAR), enable SSM for each partition.

SSM must be enabled or disabled in ALL places or errors may be logged and functions will not work as expected.

For S/390 Enterprise Servers and S/390 Multiprise servers and IBM @server zSeries, the setting of the Environment Control Mask (ECM) enables SSM. For details, refer to *Support Element Operations Guide*. For S/390 Enterprise Servers and S/390 Multiprise servers and IBM @server zSeries without TKE, the supplied ECM enables SSM. For S/390 Enterprise Servers and S/390 Multiprise processors with TKE, you can set the ECM directly; the supplied ECM enables SSM, but you have the ability to disable it. For details, refer to *z/OS ICSF TKE Workstation User's Guide 2000*.

TRACEENTRY(n)

Specifies the number, *n*, of trace buffers to allocate for ICSF tracing. Specify *n* as a decimal value from 100 through 10000, inclusive. The default is 1000.

You should set this parameter to the maximum in case you ever need this trace material.

UDX(UDX-id,service-number,load-module-name,'comment_text',FAIL(fail-option))

ICSF allows the development of User Defined Extensions for the PCI Cryptographic Coprocessor. The *UDX-id* is supplied to the installation when the UDX is developed. The *service-number* specifies a number that identifies the service to ICSF. The valid service numbers are 1 to 32767, inclusive. This set of service numbers is valid for both installation-defined services and UDX services. The service number of a UDX service must not be the same as the service number of an installation-defined service. The *load-module-name* is the name of the module that contains this service. During ICSF startup, ICSF loads this module and binds it to the service-number that was specified. A *comment* may be specified. The positional parameter is required. The comment consists of up to 40 EBCDIC characters, and may include imbedded blank characters. The comment text is enclosed by single quotes. If no comment text is desired, two contiguous single quotes should be specified.

The *fail-option* is YES or NO, indicating the action ICSF should take if loading the service ends abnormally. If the service itself ends abnormally, ICSF does not end, but takes a system dump instead.

YES Specifies that ICSF ends abnormally if your service cannot be loaded.

NO Specifies that ICSF continues to start if your service cannot be loaded.

USERPARM(value)

Specifies an 8-byte field for installation use. The Installation Option Display panel displays this value, which is stored in the Cryptographic Communication Vector Table (CCVT) in the *CCVT_USERPARM* field. An application program or installation exit can examine this field and use it to set system environment information. The default is eight blanks.

WAITLIST(data_set_name)

This optional parameter can be used if you have ICSF with CICS installed. Specifies that a customer modifiable data set will be used to determine names of the services to be placed into the ICSF CICS Wait List. A sample data set is provided by ICSF via member CSFWTL00 of SYS1.SAMPLIB. The sample data set contains the same entries as the default ICSF CICS Wait List. For example, the data set contains the names of all ICSF callable services which, by default, will be driven through the CICS TRUE. The WAITLIST option should be added to the Installation Options data set under the following conditions. For additional information on the CICS Attachment Facility, see Appendix B, "Installing the CICS-ICSF Attachment Facility", on page 181.

- Non-CICS customers will not specify a WAITLIST keyword.
- CICS customers who want to use the default CICS Wait List shipped with OS/390 V2 R10 ICSF will not specify a WAITLIST keyword. You must

ensure, however, that any existing CICS applications which invoke any of the ICSF services in the Wait List are re-linked to pick up the new version of the stub.

- CICS customers who do not want to make use of CICS TRUE must either not enable the TRUE or specify a WAITLIST keyword and point to an empty wait list data set or you can specify WAITLIST(DUMMY) in the Installation Options data set.
- CICS customers who wish to modify the ICSF default CICS Wait List should modify the sample Wait List data set supplied in member CSFWTL00 of SYS1.SAMPLIB. The WAITLIST keyword in the Installation Options Data Set should be set to point to this data set. Any existing CICS applications which invoke any of the ICSF services in the Wait List should be re-linked to pick up the new version of the stub.

Improving CKDS Performance

To improve the performance of CKDS operations during KGUP runs, use the Batch Local Shared Resource (BLSR) with Deferred Write for all KGUP runs. See *MVS Batch Local Shared Resources* for more information.

Creating ICSF Exits and Generic Services

You need not code any exits or generic services before using ICSF productively.

Developing callable service exits and generic services requires skill in assembler programming in a cross memory environment. To help with testing, the system programmer might want to use the WTO macro with the LINKAGE=BRANCH keyword to issue console messages while in cross-memory mode. (See “Callable Service Exits” on page 87 for more information.)

Chapter 3. Migration from PCF to z/OS ICSF

If your installation uses the cryptographic product, Programmed Cryptographic Facility (PCF), ICSF helps you migrate PCF applications to ICSF. You can run PCF applications on ICSF to gain the enhanced performance and availability of ICSF and to test ICSF. Eventually, you should convert these applications to use ICSF services, rather than the PCF macros.

During migration, you can run PCF applications on ICSF because ICSF continues to support the PCF macros (GENKEY, RETKEY, EMK, and CIPHER). If GENKEY or RETKEY macro exits exist, you should reevaluate their applicability to ICSF. If an exit performs a necessary function, you need to rewrite the exit for ICSF. Exits exist for the compatibility services on ICSF.

If a PCF application uses a key in the PCF cryptographic key data set, you must convert the key to an ICSF cryptographic key data set before you run the PCF application on ICSF. ICSF provides a program to make this conversion.

Running PCF and z/OS ICSF on the Same System

You can run PCF and ICSF simultaneously on the same z/OS system or separately in three different modes. You can run ICSF in compatibility, coexistence, or noncompatibility mode.

In compatibility mode, you can run either PCF or ICSF, but you cannot run them simultaneously on the same z/OS system. You can continue to run PCF applications on PCF or you can run PCF applications on ICSF. ICSF supports the PCF macros that the PCF applications call. However, you cannot run the PCF key generator utility program (KGUP) on ICSF. You do not have to reassemble PCF applications to run the applications on ICSF.

In coexistence mode, you can run PCF and ICSF simultaneously on the same z/OS system. You can continue to run a PCF application on PCF or you can reassemble the PCF application to run on ICSF. In this mode, ICSF supports the PCF macros when a reassembled PCF application calls these macros.

In noncompatibility mode, you can run PCF and ICSF simultaneously and independently on the same z/OS system. You can run PCF applications on PCF and ICSF applications on ICSF. You cannot run PCF applications on ICSF, because ICSF does not support the PCF macros in this mode.

You can run PCF simultaneously and independently in coexistence and noncompatibility mode. Therefore, in these modes, you can run PCF KGUP on PCF while running ICSF. The PCF KGUP updates keys on a PCF CKDS.

The ICSF installation option COMPAT(YES, COEXIST or NO) allows you to specify which mode you want ICSF to run in. You specify COMPAT(YES) for compatibility mode, COMPAT(COEXIST) for coexistence mode, and COMPAT(NO) for noncompatibility mode. See “Create the Installation Options Data Set” on page 14 for information about creating the installation options data set and “Changing Parameters in the Installation Options Data Set” on page 22 for details about these options.

Running in Compatibility Mode

In compatibility mode, you can run a PCF application on ICSF without reassembling the application. A PCF application running on ICSF can still use PCF macros, because ICSF supports these macros. The PCF application gains the enhanced performance, reliability, and availability of ICSF.

You cannot run PCF and ICSF simultaneously on the same z/OS system in compatibility mode. If you start PCF, you must stop PCF before you can start ICSF. If you start ICSF, you must stop ICSF before you can start PCF.

A PCF application may have used keys on the PCF cryptographic key data set (CKDS). When you run the application on ICSF, these keys must be in the ICSF CKDS. The format of a key entry on the PCF CKDS differs from the format of a key entry on the ICSF CKDS. Therefore, you need to run a conversion program to convert the PCF CKDS entries and place the entries in the ICSF CKDS. See "Converting a PCF CKDS to ICSF Format" on page 37 for a description of how to convert a PCF CKDS.

For encryption, ICSF supports the Data Encryption Standard (DES), the Commercial Data Masking Facility (CDMF), or both. Wherever possible, the key token is marked to signal which algorithm to use.

PCF macros receive identical error return codes if they run on ICSF or PCF, with one exception. If a key is installed on the ICSF CKDS with the correct label but with the wrong key type, an attempt to use that key by RETKEY or GENKEY results in a return code of 8 from PCF. This indicates that the key was not of the correct type. ICSF issues return code 12, indicating that it could not find the key. Ensure that PCF LOCAL or CROSS 1 keys are installed in the ICSF CKDS as EXPORTER keys. Also, ensure that REMOTE and CROSS 2 keys are installed in the ICSF CKDS as IMPORTER keys.

In compatibility mode, the safest method for changing the master key is to re-IPL the system. To change the master key in compatibility mode, see "Changing the Master Key in Compatibility or Coexistence Mode" on page 35.

Note: To use AMS REPRO encryption, you need to run ICSF in compatibility mode.

Running in Coexistence Mode

In coexistence mode, you can run ICSF and PCF simultaneously on the same OS/390 system and run a PCF application on PCF or on ICSF. A PCF application running on ICSF gains the enhanced performance, reliability, and availability of ICSF.

A PCF application running on ICSF can still use PCF macros, because ICSF supports these macros. ICSF ships changed PCF macros in SAMPLIB that run only on ICSF. Because these changed PCF macros already exist unchanged on PCF, the changed PCF macros shipped with ICSF are named differently.

On ICSF, in SAMPLIB:

- The changed PCF EMK macro is named CSFEMK.
- The changed PCF CIPHER macro is named CSFCIPH.
- The changed PCF RETKEY macro is named CSFRKY.
- The changed PCF GENKEY macro is named CSFGKY.

You can rename these macros to the PCF names when you want to run a PCF application on ICSF.

To run a PCF application on ICSF, you must:

- Rename the changed PCF macro shipped in ICSF SAMPLIB to the appropriate PCF name.
- Place the macro in the appropriate macro library.
- Reassemble the PCF application against the changed PCF macro.

Then the application can run only on ICSF. To run a PCF application on PCF, just run the application without reassembling the application.

During migration, you can start ICSF and start PCF so that both products are running simultaneously. If you want to run a PCF application using the PCF macros on PCF, do not reassemble the application. If you want to run a PCF application using the changed PCF macros on ICSF, reassemble the application against the changed macros. Coexistence mode enables you to run the products simultaneously and choose whether to run a PCF application on PCF or ICSF.

A PCF application can use keys on the PCF CKDS. When you run the application on ICSF, those keys must be in the ICSF CKDS. The format of a key entry on the PCF CKDS differs from the format of a key entry on the ICSF CKDS. Therefore, you need to run a conversion program to convert the PCF CKDS entries and place the entries in the ICSF CKDS. See “Converting a PCF CKDS to ICSF Format” on page 37 for a description of how to convert a PCF CKDS.

In coexistence mode, the safest method for changing the master key is to re-IPL the system. See “Changing the Master Key in Compatibility or Coexistence Mode” for a description of the process used to change the master key in coexistence mode.

Changing the Master Key in Compatibility or Coexistence Mode

In compatibility and coexistence modes, the safest way to activate a master key after changing it is to re-IPL the system. This process is different from the usual process for entering and activating a master key. For information about changing the master key, see *z/OS ICSF Administrator's Guide*.

A re-IPL ensures that a program does not access a cryptographic service with a key that is encrypted under a different master key. If a program is using an operational key, the program either re-creates the key or imports the key again.

In compatibility or coexistence mode, the ICSF administrator can use the ICSF panels to enter the key value into the new master key register. However, the master key cannot be *activated* using the panels in compatibility or coexistence mode. The value entered remains in the new master key register until you re-IPL the system. (In noncompatibility mode, the ICSF administrator can use the ICSF panels to enter the key value into the new master key register and to activate the master key.)

If the new master key is different than the current master key, the ICSF administrator must reencipher the CKDS under this new master key. To do this, choose the change option on the master key management panel. This reenciphers a CKDS under the master key in the new master key register. Reencipher all the disk copies of the CKDSs, and leave the ICSF panels without changing the master key.

Then re-IPL the system and restart ICSF. In the installation options data set, the CKDSN installation option must specify a disk copy of the CKDS that is reenciphered under the new master key. When ICSF starts again, it detects that the current master key is not the one that enciphered the CKDS that is specified in the installation options data set. ICSF detects that the CKDS is enciphered under the new master key and makes that master key active.

If your installation requires 24-hour availability and it is not possible to re-IPL the system, an alternative method is to stop all cryptographic applications, especially those using PCF macros. This helps eliminate inadvertent use of operational keys that are encrypted under the old master key. After you restart CSF, applications using an operational key can either re-create or reimport the key.

Running in Noncompatibility Mode

In noncompatibility mode PCF and ICSF can run simultaneously and independently. You can run both ICSF and PCF at the same time. Just start one and then the other. Both ICSF and PCF run completely separate from each other. Each has its own applications and each uses its own services and CKDS.

You cannot run a PCF application on ICSF, even if you reassemble it. If you run an application on ICSF that calls a PCF macro, the application ends abnormally, because ICSF does not support the PCF macros in noncompatibility mode.

Because each product runs separately, neither product loses any function in exchange for compatibility. When ICSF is in compatibility or coexistence mode, you can no longer change the master key dynamically. In noncompatibility mode, this function is still possible. Therefore, except for when your installation is migrating to ICSF, you probably want to run ICSF in noncompatibility mode.

Note: When you initialize ICSF for the first time, noncompatibility mode must be active.

Specifying Compatibility Modes during Migration

The process and duration to migrate from PCF to ICSF depend on your installation. You can use different modes in different stages of migration. To change modes, change the COMPAT option in the installation options data set and restart ICSF. When you complete migration to ICSF, you can run in noncompatibility mode to use the full function of ICSF.

When you first install an ICSF system, you can continue to run PCF for production and just test ICSF. Because you are running the products separately but simultaneously on the same z/OS system, you can run in noncompatibility or coexistence mode. To run in compatibility mode, you need more than one z/OS system. You can run the test applications on ICSF on one z/OS system while you run your production on PCF on another z/OS system.

When you begin testing ICSF, you can run existing applications in either compatibility mode or coexistence mode to test the PCF macros on ICSF. After you run the test applications, you may want to bring up production using PCF applications on ICSF. When you bring over PCF applications to ICSF, you can run in coexistence mode. In this mode, you can run an application on PCF and then reassemble the application to run the application on ICSF.

While, or after, you bring PCF applications into production on ICSF, you can run test applications that call ICSF services. You can then convert the applications that call

PCF macros to applications that call the ICSF services. The ICSF services provide enhanced key separation, performance, and function. After you convert all your PCF applications to ICSF applications, you can activate noncompatibility mode and have the full function of ICSF.

Converting a PCF CKDS to ICSF Format

During migration, you may need to convert a PCF CKDS into ICSF CKDS format if:

- PCF applications running on ICSF use keys stored in a PCF CKDS.
- Your installation uses the PCF key generator utility program to create keys and uses ICSF for other cryptographic operations. To use the keys in ICSF applications, you must convert the PCF CKDS.

ICSF provides a PCF conversion program, CSFCONV, that converts a PCF CKDS into an ICSF CKDS. The conversion program runs with certain defaults. The program converts all the entries in a PCF CKDS and converts the PCF key types into certain corresponding ICSF key types. You can use the conversion program override file to instruct the conversion program not to convert certain entries. You can also tell the conversion program to convert a PCF key type into a different ICSF key type than the default.

The following sections describe how:

- The conversion program runs with certain defaults
- To use the override file to make it run differently
- To run the conversion program

How the PCF Conversion Program Runs

You can run the PCF conversion program only after you initialize the master key and CKDS for ICSF. To run the conversion program, the CKDS you specify at ICSF startup must be initialized to contain NOCV-enablement keys. For information about master key and CKDS initialization and NOCV-enablement keys, see *z/OS ICSF Administrator's Guide*.

When the conversion program processes a PCF CKDS, the program duplicates the single length key values to create double length keys.

The conversion program merges the PCF CKDS with an input ICSF CKDS. The input ICSF CKDS is an existing disk copy of an ICSF CKDS. The input ICSF CKDS must contain a header record and include system keys entries, but may or may not contain other key entries. ICSF uses NOCV enablement keys to create keys to communicate with systems that do not use control vectors. If the ICSF CKDS resulting from the conversion contains converted importer or exporter key-encrypting keys, the input ICSF CKDS must contain NOCV enablement keys. For information about initializing an ICSF CKDS, see *z/OS ICSF Administrator's Guide*.

The PCF conversion program places the input ICSF CKDS entries and the converted PCF entries into an output CKDS. You must create an empty VSAM data set to be the output CKDS before running the conversion program. See "Create the CKDS" on page 10 for information about creating the data set.

The PCF conversion program converts all the entries in a PCF CKDS. When you run the PCF conversion program, the program does the following conversions of PCF key types into ICSF key types:

- Converts each PCF local key entry into an ICSF NOCV exporter key-encrypting key entry.
- Converts each PCF remote key entry into an ICSF NOCV importer key-encrypting key entry.
- Converts each PCF cross key entry into two ICSF key entries: an NOCV exporter key-encrypting key and an NOCV importer key-encrypting key.

You use the override file to not convert all the entries in a PCF CKDS or to convert a PCF key into a different key type than the default key type.

When the PCF conversion program converts a PCF entry, the program places any installation data from the installation data field of the PCF entry into the ICSF entry. You can use the override file to place different installation data into the ICSF entry.

Note: ICSF copies any installation data in the input CSF CKDS header record into the output ICSF CKDS header record.

As the conversion program reads the PCF CKDS, the input ICSF CKDS, and the override file, the program places key entries into a virtual image of the output ICSF CKDS. When the virtual image CKDS is complete, the conversion program reenciphers the key values of the PCF entries from under the PCF master key to under the ICSF master key. The conversion program places the reenciphered entries into the actual output CKDS.

As the conversion program creates the virtual image ICSF CKDS, the conversion program takes information from the PCF entry and possibly the override file. For each PCF entry, the conversion program checks if its key label exists in the override file. If the label does exist in the override file, the conversion program takes the action that is specified in the override file. The program either converts or bypasses the entry. If the key label does not exist in the override file, ICSF converts the entry.

The conversion program compares the converted PCF entries by label and type with the ICSF entries that already exist in the input ICSF CKDS. If there is a match, the conversion program replaces the key value from the converted entry of the PCF source into the virtual image CKDS. If there is not a match, the conversion program converts each PCF entry after checking the override file. If the label matches and the type does not, the conversion program checks to see if the type requires a unique label. If a unique label is not required, the conversion program converts the PCF entry after checking the override file. If a unique label is required, the conversion program does not convert the PCF entry and issues an error message. If the record type is DATA, DATAXLAT, MAC, MACVER, or NULL the CKDS record requires a unique label. The CKDS record also requires a unique label if the record has ever been updated by the dynamic CKDS update callable services. The conversion program also places all the input ICSF CKDS entries into the virtual image CKDS.

Calling Installation Exits During Conversion

You can call two installation exits during conversion program processing: the conversion program exit (CSFCONVX) and the single-record, read-write exit (CSFSRRW). The conversion program calls the exit at three different times: before, during, and after conversion program processing. See Chapter 7, “Installation Exits”, on page 83 for a description of the conversion program and single-record, read-write exit control blocks.

The conversion program calls the CSFCONVX exit after you submit the conversion program job, but before the program actually begins processing. At this point, you can use the exit to change the output ICSF CKDS header record installation data field.

The conversion program also calls the CSFCONVX exit during processing as the conversion program completes the virtual image ICSF CKDS, but before the conversion program reenciphers the key values. The conversion program calls the exit as it writes each record to the virtual image ICSF CKDS. At this point, you can use the exit to specify that the conversion program not place an entry into the output ICSF CKDS.

The conversion program also calls the CSFCONVX exit after the conversion program completes processing. At this point, you can use the exit to change the output ICSF CKDS header record installation data field.

As the conversion program reads the records from the virtual image ICSF CKDS to the actual output ICSF CKDS it calls the single-record, read-write exit. The conversion program calls the single-record, read-write exit as it writes each record to the output ICSF CKDS. You can use this exit to specify that the conversion program not place an entry into the output ICSF CKDS.

The conversion program writes every entry from the PCF CKDS and input ICSF CKDS into the output ICSF CKDS unless an override record or installation exit indicates that the conversion program should bypass the entry from the PCF CKDS.

Using the Conversion Program Override File

The conversion program converts all entries in a PCF CKDS into ICSF entries. The conversion program also converts each type of PCF key into a specific ICSF key type. If you want the conversion program to bypass certain key entries or convert a specific key or key type differently than it does by default, use the override file.

By specifying override records, you can have the conversion program do any of the following:

- Bypass conversion of key entries
- Include information in key entries
- Convert key types differently than it does by default

These actions can relate to entries explicitly identified with a key label or entries that are identified globally.

You specify information in certain fields in an override record and leave other fields blank, depending on the action you want the conversion program to take. You can specify a global record affecting more than one PCF CKDS entry or a record that affects only one PCF CKDS entry.

All the override data set records should be in ascending sequence by key label and old key type. If you use global entries, they must be the initial entries in the override record. Table 2 on page 40 shows the syntax of a record in the override file.

Note: All the fields should contain character values and be left-justified.

If you specify a key label in an override record, the conversion program processes the key entry identified by that key label. If you do not specify a key label in an

override record, you are using a global override record. The conversion program processes all the key labels that pertain to the information specified by the override file.

You can use a global override record to affect all the entries in a CKDS and then use override records to explicitly affect entries you did not want to have that global override record affect.

Table 2. Format of Records in the Override File

Column	Length	Description
1	8	***Key Label*** The key label of the PCF entry you want to convert The field can have the following values: • Blanks • A key label existing in the PCF CKDS that you want to convert
9	1	This field must be blank.
10	8	***Old Key Type*** The key type of the key entry you want to convert in the PCF CKDS. The field can have the following values: • Blanks • LOCAL • REMOTE
18	1	This field must be blank.
19	8	***New Key Type*** The key type that you want the converted key entry to be in the ICSF CKDS. The master key variant for the key type enciphers the key in the ICSF CKDS entry that the conversion program creates. The field can have the following values: • Blanks • OPINENC • EXPORTER • IPINENC • IMPORTER
27	1	This field must be blank.
28	8	***Ignored*** In ICSF/MVS Version 1 Release 1, this field contained the key qualifier. The CKDS for ICSF/MVS Version 1 Release 2 or above does not support key qualifiers. If your installation has a PCF conversion program override file created with ICSF/MVS Version 1 Release 1, you can still use it with z/OS ICSF. Any key qualifier entries are ignored.
36	1	This field must be blank.

Table 2. Format of Records in the Override File (continued)

Column	Length	Description
37	1	***Bypass Flag*** Used to indicate that an input CKDS entry is not to be included in the new ICSF CKDS. If you set this field to `Y`, the conversion program does not convert the entry. The field can have the following values: • Blank (same as N) • N • Y
38	1	This field must be blank.
39	52	***Installation Data*** Any additional information your installation records about a key. The information appears in the installation data field of the new ICSF CKDS. The field can contain any value.

Bypassing Conversion of Entries

Using an override record, you can bypass a PCF entry so it is not converted and placed in the ICSF CKDS. You can use a global override record to bypass all the entries in the data set and then use explicit override records to convert certain entries. You can also convert most of a PCF CKDS and just bypass certain entries using explicit override records.

Following are some examples of override records for bypassing conversion.

Example 1: This example shows an override record specifying that the conversion program not convert any PCF CKDS entry with a certain key label.

```
EXTOATM3                        Y
```

The conversion program bypasses any PCF CKDS entry with the label EXTOATM3.

Example 2: This example shows an override record specifying that the conversion program not convert any PCF CKDS entry with a certain key label and key type.

```
CRLABEL4 REMOTE                 Y
```

The conversion program bypasses any PCF CKDS entry with the label CRLABEL4 and key type REMOTE.

Example 3: This example shows a global override record specifying that the conversion program bypass all the entries in a PCF CKDS.

```
                                Y
```

The conversion program does not convert any of the entries in the PCF CKDS.

After you specify this global override record, you can use explicit override records to convert certain entries in the PCF CKDS. For example, you can use an override record like the following one to explicitly convert PCF entries with a certain label.

```
ATM03                           N
```

In this example, the conversion program converts any PCF CKDS entry with the label ATM03.

Example 4: This example shows a global override record specifying that the conversion program bypass all the entries with a certain PCF key type in a PCF CKDS.

```
REMOTE                          Y
```

The conversion program does not convert any of the entries with a key type of REMOTE in the PCF CKDS. After you specify this global override record, you can use explicit override records to convert specific entries with a key type of REMOTE in the PCF CKDS.

Including Information in a Key Entry

Programming Interface information

An ICSF key entry contains an installation data field that an installation can use to further identify a key. The installation data field contains any information that an installation wants to supply about a key.

PCF records contain an installation data field. The conversion program places the information in the field into the installation data field of the converted entry in the output ICSF CKDS. You can use an override record to specify installation data information for the converted entry in the output ICSF CKDS. The installation data information supplied in the override record overrides any information from the PCF installation data field. If you do not use an override record, the conversion program places any installation data from the PCF entry into the leftmost 8 bytes of the ICSF entry.

Following are examples of override records for including key information.

Example 1: This example shows an override record providing the conversion program with installation data information to place in the ICSF CKDS for any converted PCF entry with a certain key label.

```
ATMKEY12                        CONVERTED FROM CUSP1.CKDS 10/01/98
```

When the conversion program converts an entry that is labeled ATMKEY12, it places `CONVERTED FROM CUSP1.CKDS 10/01/98` in the installation data field for the converted entry.

Example 2: This example shows an override record providing the conversion program with installation data information to place in the ICSF CKDS for any converted PCF entry with a certain key label and key type.

```
LOCAL890 LOCAL                  CONVERTED FROM PCF12.CKDS
```

When the conversion program converts an entry that is labeled LOCAL890 with a key type of LOCAL, it places `CONVERTED FROM PCF12.CKDS` in the installation data field for the converted entry.

Example 3: This example shows a global override record that provides the conversion program with installation data information to place in the ICSF CKDS for all converted entries.

```
                                CONVERTED FROM PCF10.CKDS
```

When the conversion program converts the PCF CKDS, it places `CONVERTED FROM PCF10.CKDS` in the installation data field. The information is placed into every converted key entry. After you specify this global override record, you can use explicit override records to provide different information for specific entries in the PCF CKDS.

End of Programming Interface information

Converting Key Types

By default, the conversion program converts PCF keys into certain ICSF key types. You can use the override file to override the defaults. For example:

- Instead of automatically converting a PCF local key into a NOCV exporter key-encrypting key, you can convert the local key into an output PIN-encrypting key.
- Instead of automatically converting a PCF remote key into a NOCV importer key-encrypting key, you can convert the remote key into an input PIN-encrypting key.
- Instead of automatically converting a PCF cross key into a NOCV exporter key-encrypting key and a NOCV importer key-encrypting key, you can convert the cross key into an output PIN-encrypting key and an input PIN-encrypting key.

You can use a global override record to convert all keys of a certain type into a type other than the conversion program default key type. Then using an explicit override record, you can specify that the conversion program convert a specific record into a the default key type. For example, you can use a global override record to convert all remote keys into input PIN-encrypting keys, and then use an override record to convert specific remote entries into importer key-encrypting keys.

Following are some examples of override records for key type conversion.

Example 1: This example shows an override record specifying that the conversion program convert a local key to an output PIN-encrypting key for any PCF CKDS entry with a certain key label. The override record also provides the conversion program with installation data.

```
CRLABEL1    LOCAL  OPINENC           OPINENC FOR ATM123
```

When the conversion program converts any PCF entry labeled CRLABEL1 with a key type of local, it converts the key from a local key type to an output PIN-encrypting key type. The program also places `OPINENC FOR ATM123` in the installation data field.

If you did not specify this override record, the conversion program would automatically convert the entry from a local key type to an exporter key-encrypting key type.

Example 2: This example shows an override record specifying that the conversion program convert a remote key to an input PIN-encrypting key for any PCF CKDS entry with a certain key label. The override record also provides the conversion program with installation data.

```
CRLABEL2   REMOTE  IPINENC           IPINENC FOR ATM123
```

When the conversion program converts any PCF CKDS entry labeled CRLABEL2 with a key type of remote, it converts the key from a remote key type to an input PIN-encrypting key type. The program also places `IPINENC FOR ATM123` in the installation data field.

If you did not specify this override record, the conversion program would automatically convert the entry from a remote key type to an importer key-encrypting key type.

Example 3: This example shows an override record specifying that the conversion program convert a local key to an exporter key-encrypting key for any PCF CKDS entry with a certain key label. The override record also provides the conversion program with installation data.

```
LOLABEL1 LOCAL    EXPORTER            EXPORTER CONVERTED FROM CUSP12.CKDS
```

The conversion program automatically converts a local key to an exporter key-encrypting key. You can use this override record if you previously submitted an override record that had the conversion program convert all the local key types to output PIN-encrypting keys. You can use this override record to explicitly convert the key entry that is labeled LOLABEL1 from a local key type to an exporter key-encrypting key type.

With the example override record, when the conversion program converts any PCF entry labelled LOLABEL1 with a key type of local, it converts the key from a local key type to an exporter key-encrypting key type. The program also places `EXPORTER CONVERTED FROM CUSP12.CKDS` in the installation data field.

Example 4: This example shows an override record specifying that the conversion program convert a remote key to an importer key-encrypting key for any PCF CKDS entry with a certain key label. The override record also provides the conversion program with installation data.

```
RECKDS12 REMOTE   IMPORTER            IMPORTER CONVERTED FROM CUSP12.CKDS
```

The conversion program automatically converts remote keys to importer key-encrypting keys. You can use this override record if you supplied an override record to convert all the remote key types to input key-encrypting keys. Use this override record to explicitly convert key entries labeled RECKDS12 from remote key types to importer key-encrypting key types.

With the example override record, when the conversion program converts any PCF entry labeled RECKDS12 with a key type of remote, it converts the key from a remote key type to an importer key-encrypting key type. The program also places `IMPORTER CONVERTED FROM CUSP12.CKDS` in the installation data field.

Example 5: This example shows a global override record specifying that the conversion program convert a local key to an output PIN-encrypting key for any PCF CKDS entry with a key type of local. The override record also provides the conversion program with installation data.

```
LOCAL  OPINENC           OPINENC FROM CUSP.PIN12.CKDS
```

When the conversion program converts any PCF entry with a key type of local, the program converts the key from a local key type to an output PIN-encrypting key type. The program also places `OPINENC FROM CUSP.PIN12.CKDS` in the installation data field. After you specify this global override record, you can use explicit override records to place different installation data in the ICSF CKDS entries.

Example 6: This example shows a global override record specifying that the conversion program convert a remote key to an input PIN-encrypting key for any PCF CKDS entry with a key type of remote. The override record also provides the conversion program with installation data.

```
REMOTE  IPINENC           IPINENC FROM CUSP.PIN12.CKDS
```

When the conversion program converts any CUSP/PCF entry with a key type of remote, it converts the key from a remote key type to an input PIN-encrypting key type. The program also places `IPINENC FROM CUSP.PIN12.CKDS` in the installation data field for the entry in the ICSF CKDS. After you specify this global override record, you can use explicit override records to place different installation data information in the ICSF CKDS entries.

Running the Conversion Program

You can run the conversion program only after you initialize the master key and CKDS for ICSF. The CKDS you specify at ICSF startup must be initialized to contain NOCV-enablement keys. For information about defining keys on ICSF, see *z/OS ICSF Administrator's Guide*.

If the PCF master key and the ICSF master key are not the same, you must define the PCF master key in the input ICSF CKDS. Define the PCF master key as an importer key-encrypting key in the input ICSF CKDS. You define the key by entering the key through the key entry hardware, or by importing the key using the ICSF key generator utility program. For information about direct key entry through the key entry hardware and the key generator utility program, see *z/OS ICSF Administrator's Guide*.

Note: Be careful defining the PCF master key in the input ICSF CKDS, because there is no programmed way to determine its validity.

You run the conversion program by submitting a batch job. On the EXEC statement, specify PGM=CSFCONV. If the PCF master key and ICSF master key are not the same in the PARM= field on the EXEC statement, specify the label of the PCF master key entry in the input ICSF CKDS. If you do not specify the parameter, the conversion program assumes that the PCF master key and ICSF master key are the same.

The following example is a JCL that runs the conversion program:

```
//CKDSCONV EXEC PGM=CSFCONV,PARM='CUSPMKEY'
//CSFVSRC  DD   DSN=PROD.CUSP.CKDS,DISP=SHR
//CSFVINP  DD   DSN=TEST.CSF.CKDS,DISP=SHR
//CSFVOVR  DD   DSN=OVERRIDE.DATA,DISP=OLD
//CSFVNEW  DD   DSN=MERGED.CSF.CKDS,DISP=OLD
//CSFVRPT  DD   SYSOUT=A
//
```

In the example, CUSPMKEY is the label of the entry in the input ICSF CKDS for the PCF master key in importer key-encrypting key form. All the data sets necessary to run the conversion program are specified using DD statements.

The conversion program uses the following data sets:

CSFVSRC
: The PCF CKDS containing entries that you want to convert into ICSF format and place in the output ICSF CKDS. This is the source CKDS for the conversion. For a description of the PCF CKDS record format, see *OS/VS1 and OS/VS2 MVS Programmed Cryptographic Facility*.

CSFVINP
: A disk copy of the input ICSF CKDS. The input CKDS should already contain the header record and the ICSF system keys and can contain other ICSF key entries. If the CKDS does not contain NOCV-enablement keys,

the output ICSF CKDS will not contain NOCV-enablement keys. For more information about NOCV-enablement keys, see *z/OS ICSF Administrator's Guide*.

Note: The input ICSF CKDS does not have to be the CKDS you specify when you start ICSF.

CSFVOVR
The override file with information specifying how you want the conversion program to process PCF key entries. If no override data is required, this data set is optional. However, you must code a dummy DD statement in the JCL.

The following JCL is an example of a dummy DD statement for an override file:

```
//CSFVOVR DD DUMMY,DCB=(RECFM=FB,LRECL=90,BLKSIZE=3600)
```

See "Using the Conversion Program Override File" on page 39 for a description of when and how to use the override file.

CSFVNEW
An empty disk copy of an ICSF CKDS. This is the ICSF CKDS into which the conversion program places key entries. The conversion program places key entries from the input ICSF CKDS and the PCF CKDS into the output ICSF CKDS. The data set must be defined and empty before you run the conversion program.

CSFVRPT
The activity report that the conversion program creates. The report describes any override records and gives a summary of CKDS entries that were affected by the conversion program.

Attention: If a conversion program run ends prematurely, the results of the job are unpredictable. You should not read a CKDS involved in the conversion into storage for use. For a description of the conversion program return codes, see the explanation of message CSFV0026 in *z/OS ICSF Messages*.

When you run the conversion program, the program produces information about the conversion in an activity report. The activity report lists each override entry, the action each override entry applies to the input PCF CKDS, and any error messages. The activity report also lists the data sets that were used in the conversion and a summary of processing. The summary of processing contains totals that apply to CKDS entries in the conversion program job.

Figure 2 on page 47 is an example of an activity report with five explicit override records and no global override records.

```
CRYPTOGRAPHIC CONVERSION ACTIVITY REPORT                DATE: 2001/06/01 (YYYY/MM/DD) TIME: 10:13:09 PAGE: 1
 OVERRIDE--> CRLABEL3 LOCAL    OPINENC            Used in transfers to Main Office.
 >>>CSFV0192 TYPE FOR KEY ENTRY CRLABEL3 LOCAL CONVERTED TO OPINENC.
 >>>CSFV0232 INSTALLATION DATA FOR KEY ENTRY CRLABEL3 OPINENC SET TO Used in transfers to Main Office

 OVERRIDE--> CRLABEL3 REMOTE   IPINENC            Used in receiving from the Main Office
 >>>CSFV0192 TYPE FOR KEY ENTRY CRLABEL3 REMOTE CONVERTED TO IPINENC.
 >>>CSFV0232 INSTALLATION DATA FOR KEY ENTRY CRLABEL3 IPINENC SET TO Used in receiving from the Main Office.

 OVERRIDE--> KGLABEL1 LOCAL    OPINENC            Used for sending encrypted PINs
 >>>CSFV0292 NO KEY ENTRY FOUND FOR KGLABEL1 LOCAL.


 OVERRIDE--> LOLABEL2                           Valid for January 2001
 >>>CSFV0232 INSTALLATION DATA FOR KEY ENTRY LOLABEL2 EXPORTER SET TO Valid for January 2001.

 OVERRIDE--> ZZZZ1    LOCAL                    Y Eliminate Key from output CKDS
 >>>CSFV0382 ADD/CHANGE SPECIFICATIONS IGNORED ON OVERRIDE ENTRY. BYPASS_FLAG VALUE IS "Y".
 >>>CSFV0292 NO KEY ENTRY FOUND FOR ZZZZ1 LOCAL.


 >>>CSFV0012 CONVERSION PROCESSING COMPLETED. RETURN CODE = 4.
CRYPTOGRAPHIC CONVERSION ACTIVITY REPORT                DATE: 2001/06/01 (YYYY/MM/DD) TIME: 10:13:09 PAGE: 2

    CKDS DDNAME     Data Set Name
    ------------    --------------
    CSFVSRC         PROD.CUSP.CKDS
    CSFVINP         TEST.CSF.CKDS
    CSFVNEW         MERGED.CSF.CKDS

    PROCESSING SUMMARY

             Source CKDS Entries               Converted Entries                    ICSF Entries
             --------------------------------  ----------------------------------   ----------------------------------
                LOCAL                     4    * Candidates                  16     + Changed Input Entries          2
                REMOTE                    4      Bypassed by Overrides      ( 0)      Unchanged Input Entries       13
                CROSS                     4                                          --------------------------------
                ------------------------------   --------------------------------     TOTAL ICSF Input Entries     15
              * TOTAL Source Entries      12     TOTAL Converted Entries      16    + Entries Added from Source    14
                                                                                      Entries Bypassed by Exit    ( 0)
                                                                                      --------------------------------
                                                                                      TOTAL Output ICSF Entries    29


  * One Source CKDS CROSS entry converts to two Candidates.
  + Total Converted Entries = Changed Input Entries + Entries Added from Source.
```

Figure 2. Example of a Conversion Initial Activity Report

In the report, the first override record specifies that when the conversion program converts a PCF entry labeled CRLABEL3 with a key type of local, the program should convert the entry into an output PIN-encrypting key. The conversion program also places the information `Used in transfers to Main Office` in the installation data field of the output ICSF CKDS entry.

The second override record specifies that when the conversion program converts a PCF entry labeled CRLABEL3 with a key type of remote, the program should convert the key into an input PIN-encrypting key. The conversion program places the information `Used in receiving from the Main Office` in the installation data field of the output ICSF CKDS entry.

The label specified by the third override record does not exist in the PCF CKDS. Therefore, the conversion program ignores this override record.

The fourth override record specifies that when the conversion program converts a PCF entry labelled LOLABEL2, the program should place the information `Valid for January 2001` in the installation data field of the output ICSF CKDS record.

The label specified by the fifth override record does not exist on the PCF CKDS that the conversion program is converting. Therefore, the conversion program ignores this override record.

The message that the conversion processing has been completed is followed by a return code. Return codes are documented under message CSFV0026 in *z/OS ICSF Messages*.

After describing the five override records, the conversion report lists the data sets the conversion program used in the conversion. PROD.CUSP.CKDS is the PCF CKDS that the program converted. TEST.CSF.CKDS is the input ICSF CKDS containing the ICSF entries input during the conversion. MERGED.CSF.CKDS is the output ICSF CKDS where the conversion program placed the converted entries.

Then the activity report lists totals pertaining to the conversion. The PCF CKDS has a total of 12 entries: four with a key type of local, four with a key type of remote, and four with a key type of cross. Because the conversion of each cross key entry results in two ICSF entries, the total ICSF entries that are candidates for conversion from the PCF is 16. None of these candidates was bypassed because of an override record, so 16 PCF entries were converted.

There were 15 entries in the input ICSF CKDS, and two of these entries were updated because they had identical key labels in the PCF CKDS. Fourteen new output ICSF CKDS entries were added from the PCF CKDS. The total number of entries in the output ICSF CKDS is 29. This includes the 15 entries in the input ICSF CKDS and the 14 entries added from the PCF CKDSN. No entries were bypassed because of the conversion program exit.

Figure 3 on page 49 is an example of an activity report with a global override record that has the conversion program bypass all the entries in the PCF CKDS. Then two override records are used to convert specific entries.

```
CRYPTOGRAPHIC CONVERSION ACTIVITY REPORT                DATE: 2001/06/01 (YYYY/MM/DD) TIME: 10:13:09 PAGE: 1
OVERRIDE-->                                     Y
 >>>CSFV0172 ALL  ENTRIES BYPASSED.

 OVERRIDE--> CRLABEL3 LOCAL    OPINENC             Used in transfers to Main Office
 >>>CSFV0222 KEY ENTRY CRLABEL3 LOCAL NOT BYPASSED.
 >>>CSFV0192 TYPE FOR KEY ENTRY CRLABEL3 LOCAL CONVERTED TO OPINENC.
 >>>CSFV0232 INSTALLATION DATA FOR KEY ENTRY CRLABEL3 OPINENC SET TO Used in transfers to Main Office.

OVERRIDE--> LOLABEL2                            Valid for January 2001
 >>>CSFV0222 KEY ENTRY LOLABEL2 LOCAL NOT BYPASSED.
 >>>CSFV0232 INSTALLATION DATA FOR KEY ENTRY LOLABEL2 EXPORTER SET TO Valid for January 2001.


 >>>CSFV0012 CONVERSION PROCESSING COMPLETED.  RETURN CODE = 0.

CRYPTOGRAPHIC CONVERSION ACTIVITY REPORT                DATE: 2001/06/01 (YYYY/MM/DD) TIME: 10:13:09 PAGE: 2

    CKDS DDNAME     Data Set Name
    ------------    --------------
    CSFVSRC         PROD.PCF.CKDS
    CSFVINP         INTEST.CSF.CKDS
    CSFVNEW         NEWTEST.CSF.CKDS

    PROCESSING SUMMARY

             Source CKDS Entries               Converted Entries                    ICSF Entries
             --------------------------------  ---------------------------------    ---------------------------------
                LOCAL                     4    * Candidates                   16     + Changed Input Entries         1
                REMOTE                    4      Bypassed by Overrides       (14)      Unchanged Input Entries      27
                CROSS                     4                                           ---------------------------------
                ------------------------------     -------------------------------    TOTAL ICSF Input Entries     28
              * TOTAL Source Entries      12       TOTAL Converted Entries       2    + Entries Added from Source     1
                                                                                       Entries Bypassed by Exit    ( 0)
                                                                                       ---------------------------------
                                                                                       TOTAL Output ICSF Entries    29


  * One Source CKDS CROSS entry converts to two Candidates.
  + Total Converted Entries = Changed Input Entries + Entries Added from Source.
```

Figure 3. Example of a Conversion Update Activity Report

The first override record specifies that the conversion program bypass all the entries in the PCF CKDS. The second override record specifies that the conversion program convert a PCF entry labeled CRLABEL3 with a key type of local into an output PIN-encrypting key. This second override record also instructs the conversion program to place the phrase `Used in transfers to Main Office` in the installation data field of the output ICSF CKDS entry. The third override record specifies that the conversion program convert a PCF entry labeled LOLABEL2 and place `Valid for January 2001` in the installation data field of the output ICSF CKDS entry.

After describing the three override records, the conversion report lists the data sets the conversion program used in the conversion. PROD.PCF.CKDS is the PCF CKDS that the program converted. INTEST.CSF.CKDS is the input ICSF CKDS that contains the ICSF entries input containing the ICSF entries input during the conversion. NEWTEST.CSF.CKDS is the output ICSF CKDS where the conversion program placed the converted entries.

Then the activity report lists totals pertaining to the conversion. The PCF CKDS has a total of 12 entries: four with a key type of local, four with a key type of remote, and four with a key type of cross. Because the conversion of each cross key entry results in two ICSF entries, the total ICSF records that are candidates for conversion from PCF is 16. Fourteen of those 16 entries were bypassed because of the global override record.

There were 28 entries in the input ICSF CKDS, and one of these entries was updated because it had an identical key label in the PCF CKDS. The total number

of entries in the output ICSF CKDS is 29. This includes the 28 entries in the input ICSF CKDS plus the one added from the PCF CKDS. No entries were bypassed because of the conversion program exit.

Chapter 4. Migration from Previous Releases of ICSF

This chapter describes migration considerations.

Your plan for migrating to the new level of ICSF should include information from a variety of sources. These sources of information describe topics such as coexistence, service, hardware and software requirements, installation and migration procedures, and interface changes.

The following documentation, which is supplied with your product order, provides information about installing your z/OS system. In addition to specific information about ICSF, this documentation contains information about all of the z/OS elements.

- *z/OS and z/OS.e Planning for Installation*

  This document describes the installation requirements for z/OS at a system and element level. It includes hardware, software, and service requirements for both the driving and target systems. It also describes any coexistence considerations and actions.
- *z/OS Program Directory*

  This document, which is provided with your z/OS product order, leads you through the specific installation steps for ICSF and the other z/OS elements.
- *ServerPac Installing Your Order*

  This is the order-customized, installation document for using the ServerPac Installation method. Be sure to review “Appendix A. Product Information”, which describes data sets supplied, jobs or procedures that have been completed for you, and product status. IBM may have run jobs or made updates to PARMLIB or other system control data sets. These updates could affect your migration.

Terminology

This section describes some terms you may need to know as you use this document.

Migration — Activities that relate to the installation of a new version or release of a program to replace an earlier level. Completion of these activities ensures that the applications and resources on your system will function correctly at the new level.

Coexistence — Two or more systems at different levels (for example, software, service or operational levels) that share resources. Coexistence includes the ability of a system to respond in the following ways to a new function that was introduced on another system with which it shares resources: ignore a new function, terminate gracefully, support a new function. The following are examples of multisystem configurations in which resource sharing can occur:

- A single system running multiple LPARs
- A single processor that is time-sliced to run different levels of the system (for example, during different times of the day)
- Two or more systems running separate processors

- A Parallel Sysplex configuration (also includes a basic sysplex)

Common Migration Activities for z/OS ICSF, OS/390 ICSF and ICSF/MVS Version 2 Release 1

The following sections describe common activities and considerations that should be considered when you migrate from:

- z/OS V1 R1 and higher
- OS/390 V2 R6 ICSF and higher
- OS/390 V2 R4 ICSF
- ICSF/MVS Version 2 Release 1

Attention

If you are migrating to HCR7708 and are running on an IBM @server zSeries 990, see Appendix E, "Running HCR7708 on a IBM @server zSeries 990", on page 195 for migration and coexistence actions.

If you are migrating to HCR7708 and are running on an IBM @server zSeries 900 or below, the following migration and coexistence actions are applicable.

For a list of specific OS/390 V2 R4 migration activities, see "Migrating from OS/390 V2 R4 ICSF" on page 61. For a list of specific ICSF/MVS Version 2 Release 1 migration activities, see "Migrating from ICSF/MVS Version 2 Release 1" on page 61.

Access to Callable Services

Access to services that are executed on the PCI Cryptographic Coprocessor is through Access Control Points in the DEFAULT Role. To execute callable services on the PCI Cryptographic Coprocessor, access control points must be enabled for each service in the DEFAULT Role. The ability to enable/disable access control points in the DEFAULT Role was introduced on OS/390 V2 R10 through APAR OW46381 for the Trusted Key Entry Workstation. For systems that do not use the optional TKE Workstation, all access control points (current and new) are enabled in the DEFAULT Role with the appropriate microcode level on the PCI Cryptographic Coprocessor. New TKE users and non-TKE users have all access control points enabled. This is also true for brand new TKE V3.1 users (not converting from TKE V3.0).

Note: Access control point DKYGENKY-DALL is always disabled in the DEFAULT Role for all customers (TKE and Non-TKE). A TKE Workstation is required to enable this access control point for the Diversified Key Generate service.

For existing TKE V3.0 users, upgrading to TKE V3.1 (APAR OW46381 and its corresponding ECA), current access control points in the DEFAULT Role are enabled. Any new access control points are disabled in the DEFAULT Role and must be enabled through TKE if the service is required.

Notes:

1. APAR OW46381 will update the TKE Host Code (z/OS V1 R1 and below)
2. ECA 186 will update the TKE Workstation Code

3. The latest or most current driver is required for the PCI Cryptographic Coprocessor microcode for the S/390 G5 Enterprise Server or the S/390 G6 Enterprise Server
4. The latest or most current driver is required for the PCI Cryptographic Coprocessor microcode for the IBM @server zSeries 900

All of the above components are required for complete access control point support.

Access to services which execute on the Cryptographic Coprocessor Feature is through SAF. Disablement through SAF is sufficient to prevent execution of a service by either the Cryptographic Coprocessor Feature or the PCI Cryptographic Coprocessor. For functions which can be executed on the PCI Cryptographic Coprocessor, enablement of the function requires that the function be enabled through SAF and through the access control point in the DEFAULT Role.

If you are on OS/390 V2 R10, using a TKE V3.0 workstation, access control points for new services (requiring APARs OW46380 and OW46382) will be disabled. Existing access control points will be enabled in the DEFAULT Role. APAR OW46381 must be installed to enable the OS/390 V2 R10 interface. This will allow the TKE Administrator to enable any new access control points for ICSF services that execute in the PCI Cryptographic Coprocessor under the DEFAULT Role.

Access Control Points (requiring APARs OW46380 and OW46382) for OS/390 V2 R10 are:

- DATAM Key Management Control

  Note: For existing TKE installations (upgrading to TKE V3.1), it is required that this access control point be enabled. Failure to do so will result in processing errors for Double MAC keys in Key Import, Key Export, and Key Generate.
- Diversified Key Generate - Single length or same halves
- Diversified Key Generate - CLR8-ENC
- Diversified Key Generate - TDES-ENC
- Diversified Key Generate - TDES-DEC
- Diversified Key Generate - SESS-XOR
- Diversified Key Generate - DKYGENKY-DALL

  Note: This access control point is always disabled in the DEFAULT Role for all customers (TKE and Non-TKE). A TKE Workstation is required to enable the function.
- MAC Generate - For existing TKE installations, it is recommended that this access control point be enabled.
- MAC Verify - For existing TKE installations, it is recommended that this access control point be enabled.

Access Control Points for z/OS V1 R2 are:

- PKA Key Token Change
- Secure Messaging for Keys
- Secure Messaging for PINs

Access Control Points for z/OS V1 R3 are:

- UKPT - PIN Verify, PIN Translate

Access Control Points for z/OS V1 R4 are:

- Data Key Export - Unrestricted
- Data Key Import - Unrestricted
- Key Export - Unrestricted
- Key Import - Unrestricted
- Key Part Import - Unrestricted

These access control points were added with APAR OW53666 for HCR7703, HCR7704, and HCR7708. They are in the base for HCR7708 so the APAR doesn't need to be installed. If the customer has TKE 3.0 or 3.1 and did not have the APAR installed on one of the previous levels, the access control points would need to be enabled to allow the services to work as they did in the previous release. If the APAR was installed, the state of the access control points would be saved and the customer would not need to do anything.

Callable Services

Attention

If you are running on a IBM @server zSeries 990, see "Callable services" on page 196 for the callable services that are available to you.

- Clear PIN Generate Alternate (CSNBCPA) - If you specify a PIN extraction keyword for a given PIN block format in *rule_array*, apply APAR OW52383 on HCR7703 (OS/390 V2 R10 and z/OS V1 R1), HCR7704 (z/OS V1 R2), and HCR7705 (z/OS V1 R3). If you specify a PIN extraction keyword that is the default for the PIN block format, the request will be processed on the Cryptographic Coprocessor Feature. If the PIN extraction keyword is not the default for the PIN block format, the request will be routed to a PCI Cryptographic Coprocessor.
- Control Vector Generate (CSNBCVG) - Beginning in OS/390 V2 R10, this callable service has been enhanced to support new key types KEYGENKY, DKYGENKY, and SECMSG. The following *rule_array* keywords are also supported: CLR8-ENC, DALL, DDATA, DEXP, DIMP, DKYL0, DKYL1, DKYL2, DKYL3, DKYL4, DKYL5, DKYL6, DKYL7, DMAC, DMKEY, DMPIN, DMV, DPVR, SMKEY, and SMPIN.

  Beginning in z/OS V1 R3, the *rule_array* parameter has been enhanced to support the UKPT keyword.
- Digital Signature Generate (CSNDDSG) - Beginning with OS/390 V2 R9 ICSF, if you specify ZERO-PAD in the *rule_array parameter*, the input hash length is limited to 32 bytes (256 bits). APAR OW48511 (for OS/390 V2 R9 and OS/390 V2 R10) changes the hash length limit to 256 bytes when ZERO-PAD is specified for signature use only keys. It also increases the hash length limit for all other keys when ZERO-PAD is specified to 36 bytes.

  Beginning in OS/390 V2 R10, ANSI X9.31 formatting for a digital signature is supported. New *rule_array* keywords are: X9.31, SHA-1, and RPMD-160.
- Digital Signature Verify (CSNDDSV) - Beginning in OS/390 V2 R10, ANSI X9.31 formatting for a digital signature is supported. New *rule_array keyword* X9.31 has been added.
- Diversified Key Generate (CSNBDKG) - This is a new service in OS/390 V2 R10. This service generates a key based on the key-generating key, the processing method, and the parameter supplied.

- Encrypted PIN Translate (CSNBPTR) - Beginning in z/OS V1 R3, the *rule_array* parameter has been enhanced to support UKPT keywords UKPTIPIN, UKPTOPIN, and UKPTBOTH.

  If you specify a PIN extraction keyword for a given PIN block format in *rule_array*, apply APAR OW52383 on HCR7703 (OS/390 V2 R10 and z/OS V1 R1), HCR7704 (z/OS V1 R2), and HCR7705 (z/OS V1 R3).If you specify a PIN extraction keyword that is the default for the PIN block format, the request will be processed on the Cryptographic Coprocessor Feature. If the PIN extraction keyword is not the default for the PIN block format, the request will be routed to a PCI Cryptographic Coprocessor..
- Encrypted PIN Verify (CSNBPVR) - Beginning in z/OS V1 R3, the *rule_array* parameter has been enhanced to support the UKPT keyword UKPTIPIN.

  If you specify a PIN extraction keyword for a given PIN block format in *rule_array*, apply APAR OW52383 on HCR7703 (OS/390 V2 R10 and z/OS V1 R1), HCR7704 (z/OS V1 R2), and HCR7705 (z/OS V1 R3). If you specify a PIN extraction keyword that is the default for the PIN block format, the request will be processed on the Cryptographic Coprocessor Feature. If the PIN extraction keyword is not the default for the PIN block format, the request will be routed to a PCI Cryptographic Coprocessor.

  Rule_array has been enhanced to support the VISAPVV4 keyword in HCR7708. This function is rolled back to V1 R3 and V1 R2 when APAR OA02575 is applied.
- Key Generate (CSNBKGN) - Beginning in OS/390 V2 R10, this callable service has been enhanced to support KEYGENKY and DKYGENKY key types through the TOKEN key type keyword and the specification of the proper control vector in the *target_key_identifier* field.
- Key Export (CSNBKEX) - Beginning in OS/390 V2 R10, this callable service has been enhanced to support the source key being specified as a label.
- Key Token Build (CSNBKTB) - Beginning in OS/390 V2 R10, this callable service has been enhanced to support new key types: KEYGENKY, DKYGENKY, and SECMSG.

  Beginning in z/OS V1 R3, the *rule_array* parameter has been enhanced to support the UKPT keyword.
- MAC Generate (CSNBMGN) - beginning in HCR7708, this service has been enhanced to support longer text on a PCI Cryptographic Coprocessor.
- MAC Verify (CSNBMVR) - beginning in HCR7708, this service has been enhanced to support longer messages on a PCI Cryptographic Coprocessor.
- One-Way Hash Generate (CSNBOWH) - Beginning in OS/390 V2 R10, this callable service has been enhanced to support the RIPEMD-160 hash algorithm.
- PCI Interface (CSFPCI) - Beginning in OS/390 V2 R10, this callable service has been enhanced to query a list of enabled/disabled access control points.
- PKA Decrypt (CSNDPKD) - Beginning in OS/390 V2 R12, this callable service has been enhanced to support a clear RSA modulus-exponent or Chinese Remainder key. With APAR OW46379, the support is available in OS/390 V2 R9 and OS/390 V2 R10.

  Beginning in z/OS V1 R2, this service exploits the PCI accelerator for clear keys.
- PKA Encrypt (CSNDPKE) - A new *rule_array* parameter, ZERO-PAD, has been added in OS/390 R2 R10 (APAR OW48132). The key value will be padded on the left with binary zeros to the length of the PKA key modulus.
- PKA Key Generate (CSNDPKG) - Beginning with OS/390 V2 R9 ICSF, CSNDPKG supports writing the *generated_key* directly to the PKDS. This means that the *generated_key_token* field is now an INPUT as well as an OUTPUT field.

If a PKDS label name is not being supplied, then a value less than a blank X'40' must be supplied in the first byte of the parameter or else the service fails with a return code 8 reason code X'2AF8'.

Beginning in OS/390 V2 R10, this service was enhanced to support the XPORT *rule_array* parameter.

- PKA Key Import (CSNDPKI) - Beginning with OS/390 V2 R9 ICSF, CSNDPKI supports writing the *target_key_identifier* directly to the PKDS. This means that the *target_key_identifier* field is now an INPUT as well as an OUTPUT field. If a PKDS label name is not being supplied, then a value less than a blank X'40' must be supplied in the first byte of the parameter or else the service fails with a return code 8 reason code X'2AF8'.
- PKA Key Token Change (CSNDKTC) - This service is new in z/OS V1 R2. It changes PKA private key tokens (RSA and DSS) from encipherment with the old PCI Cryptographic Coprocessor ASYM-MK to encipherment with the current PCI Cryptographic Coprocessor ASYM-MK. PKA private keys encrypted under the KMMK cannot be reenciphered using this service unless the KMMK has the same value as the SMK.
- Public Key Extract (CSNDPKX) - Beginning with OS/390 V2 R9 ICSF, this service must be in task mode, not SRB mode. It was also enhanced to support PKDS labels as well as tokens. This requires a change to the stub module CSNDPKX. Existing applications that have been link edited with the old stub module will still run without change. Access to this service can also be RACF controlled.
- Secure Messaging for Keys (CSNBSKY) - This is a new service for z/OS V1 R2. It encrypts a text block, including a clear key value decrypted from an internal or external DES token.
- Secure Messaging for PINs (CSNBSPN) - This is a new service for z/OS V1 R2. It encrypts a text block, including a clear PIN block recovered from an encrypted PIN block. The clear PIN block can be self encrypted before it is included in the text block.
- Symmetric Key Decipher (CSNBSYD) - This is a new service for z/OS V1 R3. This callable service deciphers data in an address space using the cipher block chaining or electronic code book modes. The Symmetric Key Decipher service (AES support) is also available on z/OS V1 R2 through APAR OW52813.

  Beginning in HCR7708, this service has been enhanced to support the DES and TDES algorithms on a IBM @server zSeries 990. *Rule_array* key processing rules CUSP, IPS, and X9.23 have been added.

  Symmetric Key Decipher (CSNBSYD1) ALET support added in HCR7708.
- Symmetric Key Encipher (CSNBSYE) - This is a new service for z/OS V1 R3. This callable service enciphers data in an address space using the cipher block chaining or electronic code book modes. The Symmetric Key Encipher service (AES support) is also available on z/OS V1 R2 through APAR OW52813.

  Beginning in HCR7708, this service has been enhanced to support the DES and TDES algorithms. *Rule_array* key processing rules CUSP, IPS, and X9.23 has been added.

  Symmetric Key Encipher (CSNBSYE1) ALET support added in HCR7708.
- Symmetric Key Export (CSNDSYX) - Beginning in z/OS V1 R3, a new *rule_array* keyword, PKCSOAEP, has been added. This keyword specifies the method found in RSA PKCS #1V2 OAEP. APAR OW50507 is available on HCR7703 (OS/390 V2 R10 and z/OS V1 R1) and HCR7704 (z/OS V1 R2).
- Symmetric Key Generate (CSNDSYG) - Beginning in z/OS V1 R3, a new *rule_array* keyword, PKCSOAEP, has been added. This keyword specifies the

method found in RSA PKCS #1V2 OAEP. APAR OW50507 is available on HCR7703 (OS/390 V2 R10 and z/OS V1 R1) and HCR7704 (z/OS V1 R2).

- Symmetric Key Import (CSNDSYI) - Beginning with OS/390 V2 R9 ICSF, the *target_key_identifier_length* parameter size must be 64 bytes.

  Beginning in z/OS V1 R3, a new *rule_array* keyword, PKCSOAEP, has been added. This keyword specifies the method found in RSA PKCS #1V2 OAEP. APAR OW50507 is available on HCR7703 (OS/390 V2 R10 and z/OS V1 R1) and HCR7704 (z/OS V1 R2).
- Beginning in z/OS V1 R2, MAXLEN parameter checking has been eliminated for the following services:
  - Encipher (CSNBENC and CSNBENC1)
  - Decipher (CSNBDEC and CSNBDEC1)
  - MAC generate (CSNBMGN and CSNBMGN1)
  - MAC verify (CSNBMVR and CSNBMVR1)
  - Ciphertext translate (CSNBCTT and CSNBCTT1)
  - MDC generate (CSNBMDG and CSNBMDG1)

  The MAXLEN parameter is also no longer enforced in the CUSP compatibility CIPHER service. The MAXLEN parameter may still be specified in the options data set, but only the maximum value limit will be enforced (2147483647). If a value greater than this is specified, an error will result and ICSF will not start.

CICS Attachment Facility

If you have the CICS Attachment Facility installed with OS/390 V2 R4 ICSF and above (including z/OS), and you have ICSF CICS TRUE enabled, install APAR OW40011 on HCRP210 and APAR OW43444 on HCRP220 and HCRP230 and relink your applications that invoke the following ICSF services to pick up the updated service stubs.

Note: If you have previously installed these APARs and relinked your applications at that time, no action is required.

- HCRP210 (ICSF/MVS V2 R1, OS/390 V2 R4 ICSF, OS/390 V2 R5 ICSF): CSNBKRC, CSNBKRW, CSNBKRD, CSNDDSG, CSNDDSV, CSNDPKG, CSNDPKI, CSNDSYX, CSNDSYG, CSNDSYI, CSNDKRC, CSNDKRW, CSNDKRD, CSNDKRR, CSNDSBC, CSNDSBD
- HCRP220 (OS/390 V2 R6 ICSF, OS/390 V2 R7 ICSF, OS/390 V2 R8 ICSF): CSNBKRC, CSNBKRW, CSNBKRD, CSNDDSG, CSNDDSV, CSNDPKG, CSNDPKI, CSNDSYX, CSNDSYG, CSNDSYI, CSNDKRC, CSNDKRW, CSNDKRD, CSNDKRR, CSNDSBC, CSNDSBD, CSNDPKD, CSNDPKE
- HCRP230 (OS/390 V2 R9 ICSF): CSNBKRC, CSNBKRW, CSNBKRD, CSNDDSG, CSNDDSV, CSNDPKG, CSNDPKI, CSNDSYX, CSNDSYG, CSNDSYI, CSNDKRC, CSNDKRW, CSNDKRD, CSNDKRR, CSNDSBC, CSNDSBD, CSNDPKD, CSNDPKE, CSNDPKX, CSNDRKD, CSNDRKL

If you are running with the ICSF-CICS Attachment Facility enabled on HCR7703 (OS/390 V2 R10), HCR7704 (z/OS V1 R2), or HCR7706 (z/OS V1 R3), and have CICS transactions which invoke cryptographic services in 24 bit mode, install APAR OW53108. This APAR stores the ICSF CICS waitlist below the 16 meg line. No RE-IPL or relink is necessary. Issue the command F LLA, REFRESH and stop and restart ICSF to apply this APAR.

The value of TALENGTH has changed from 64 to 140. If you are running OS/390 R10, z/OS R1 or z/OS R2, the change can be applied with APAR OW55631. If you are running OS/390 R9, the change can be applied with APAR OW57158.

CKDS

If you are migrating from ICSF/MVS Version 2 Release 1, see “Migrating from OS/390 V2 R4 ICSF” on page 61 for a specific CKDS migration information. The following applies if you are migrating from OS/390 V2 R4 ICSF or OS/390 V2 R6 and higher ICSF.

Restriction: FMID HCR7708 running on an IBM @server zSeries 990 does not support the CKDS.

Once new key types are added to the CKDS, the following considerations apply when sharing the CKDS with non-R10 or non-z/OS systems:

- once keys with non-CCF control vectors are added to the CKDS, a CKDS reencipher operation must be invoked from a system which has a PCI Cryptographic Coprocessor installed.
- once keys of type IMPORTER, EXPORTER, PINGEN, PINVER, IPINENC, or OPINENC which have non-CCF control vectors are added to the CKDS, a toleration APAR OW43926 must be installed on the non-OS/390 V2 R10 ICSF systems. The APAR ensures that ICSF services will fail a request to use a key which contains a non-CCF control vector.

Installation Options Data Set

PKDSCACHE, an installation option, defines the size of the PKDS Cache in records. The PKDS cache improves performance as it facilitates access to frequently used records. Specify *n* as a decimal value from 0 to 256. If *n* is zero, no cache will be implemented. If PKDSCACHE is not specified, the default value is 64. PKDSCACHE can be implemented on OS/390 V2 R10 and z/OS V1 R1 by installing APAR OW48568.

Key Tokens

- Existing DES internal key tokens can be used on either the Cryptographic Coprocessor Feature or the PCI Cryptographic Coprocessor.
- An existing PKA internal token created for the Cryptographic Coprocessor Feature cannot be used on the PCI Cryptographic Coprocessor unless you recreate it by reimporting the key. Since the Cryptographic Coprocessor Feature cannot generate PKA keys (these tokens were all generated on another platform and imported for use with the Cryptographic Coprocessor Feature), you’ll need to reimport them to use them on the PCI Cryptographic Coprocessor. For maximum flexibility, you should install the SMK to be equal to the KMMK. Existing PKA tokens should then be reimported.

PCI Cryptographic Accelerator

If you have a PCI Cryptographic Accelerator online, toleration APAR OW49402 is required on lower levels of ICSF (OS/390 V2 R9, OS/390 V2 R10 and z/OS V1 R1). Without this APAR, ICSF will abend with an X'18F' reason code 50.

PKA Public Key Storage

You need to create a PKDS in order to do the following:

- Start V2 R9 ICSF or higher.

- Use the PKDS update callable services.

  These callable services will not work with any public key storage mechanisms other than the PKDS. Therefore, you will need to migrate any existing public keys to the PKDS.
- Use key labels for PKA keys instead of tokens on certain callable services.
- A PKDS is required to generate and use RSA private keys that are retained within a PCI Cryptographic Coprocessor.

PKDS

Beginning with OS/390 V2 R9, the PKDS is required.

Restriction: FMID HCR7708 running on an IBM @server zSeries 990 does not support the PKDS.

Beginning in z/OS V1 R2, support to REENCIPHER PKDS and ACTIVATE PKDS has been added to the Master Key Management Panels and to the new CSFPUTIL utility. CSFPUTIL is a new utility that performs the same functions as REENCIPHER PKDS and ACTIVATE PKDS. These functions allow you to reencipher the PKDS from the old asymmetric-keys master key to the current master key and activate the reenciphered PKDS. Other systems with lower levels of ICSF which are sharing the PKDS would disable PKDS read and PKDS write and activate the reeniciphered PKDS. For information on managing and sharing the PKDS in a sysplex environment, see *z/OS ICSF Administrator's Guide*, SA22-7521. Toleration APAR OW49386 is required on the following systems in order to activate the re-enciphered PKDS:

- HCRP210 (standalone), HCRP220 (OS/390 V2 R6, OS/390 V2 R7, OS/390 V2 R8), HCRP230 (OS/390 V2 R9), and HCR7703 (OS/390 V2 R10 and z/OS V1 R1)

With OS/390 V2 R6 ICSF and above (including z/OS), if you share the PKDS with lower level releases of ICSF, the following APARS must be installed:

- HCRP210 (ICSF/MVS V2 R1, OS/390 V2 R4 ICSF, OS/390 V2 R5 ICSF) must have APARS OW33234 and OW37623 installed. If you are running OS/390 V2 R9 ICSF, you must also have APAR OW43275 installed on HCRP210. New OS/390 V2 R9 ICSF tokens on previous releases of ICSF will be handled as follows. For additional information on ME and CRT tokens, see diagnosis reference information in *z/OS ICSF System Programmer's Guide*.
  - Retained key tokens contain a public key token. These public key tokens may be used in public key services such as Digital Signature Verify (CSNDDSV). These tokens may not be updated or deleted through the PKDS Record Write (CSNDKRW) or PKDS Record Delete (CSNDKRD) callable services.
  - All modulus-exponent form RSA internal key tokens imported or created on an OS/390 V2 R9 ICSF or OS/390 V2 R10 ICSF system with PCI Cryptographic Coprocessor will have a private section identifier of X'06'. These tokens will be converted where possible to internal tokens with a private section of X'02' for use on previous levels of ICSF without a PCI Cryptographic Coprocessor. Modulus-exponent tokens with a private section identifier of X'06' which are signature-use only tokens can be converted since these tokens are encrypted under the ASYM-MK of the PCI Cryptographic Coprocessor (which is the same as the SMK of the Cryptographic Coprocessor Feature). Modulus-exponent tokens with a private section identifier of X'06' which are designated as key-management usage can only be converted for use on previous levels of ICSF if the KMMK on the Cryptographic Coprocessor Feature is the same as the SMK.

  - CRT tokens are not supported on previous levels of ICSF.
- HCRP220 (OS/390 V2 R6 ICSF, OS/390 V2 R7 ICSF, OS/390 V2 R8 ICSF) must have APAR OW37623 installed. If you are running OS/390 V2 R9 ICSF, you must also have APAR OW43275 installed on HCRP220. New OS/390 V2 R9 ICSF tokens on previous releases of ICSF will be handled as follows. For additional information on ME and CRT tokens, see diagnosis reference information in *z/OS ICSF System Programmer's Guide*.
  - Retained key tokens contain a public key token. These public key tokens may be used in public key services such as Digital Signature Verify (CSNDDSV). These tokens may not be updated or deleted through the PKDS Record Write (CSNDKRW) or PKDS Record Delete (CSNDKRD) callable services.
  - All modulus-exponent form RSA internal key tokens imported or created on an OS/390 V2 R9 ICSF or OS/390 V2 R9 ICSF system with PCI Cryptographic Coprocessor will have a private section identifier of X'06'. These tokens will be converted where possible to internal tokens with a private section of X'02' for use on previous levels of ICSF without a PCI Cryptographic Coprocessor. Modulus-exponent tokens with a private section identifier of X'06' which are signature-use only tokens can be converted since these tokens are encrypted under the ASYM-MK of the PCI Cryptographic Coprocessor (which is the same as the SMK of the Cryptographic Coprocessor Feature). Modulus-exponent tokens with a private section identifier of X'06' which are designated as key-management usage can only be converted for use on previous levels of ICSF if the KMMK on the Cryptographic Coprocessor Feature is the same as the SMK.
  - CRT tokens are not supported on previous levels of ICSF.

Resource Manager Interface (RMF)

Beginning in z/OS V1 R3, support to enable RMF to provide performance measurements on the following selected ICSF services and functions that use Direct Access Crypto (DAC) CCF instructions has been added. Support to enable RMF is also available on z/OS V1 R2 through APAR OW51003.

- Encipher (CSNBENC)
- Decipher (CSNBDEC)
- MAC Generate (CSNBMGN)
- MAC Verify (CSNBMVR)
- One-Way Hash (CSNBOWH)
- PIN Translate (CSNBPTR)
- PIN Verify (CSNBPVR)

Special Secure Mode

Use of some ICSF services (CSNBSKI, CSNBSKM, CSNBPGN, CSNDSYG with the IM keyword) requires that ICSF be in special secure mode.

Note: If a PCI Cryptographic Coprocessor is available and the modulus bit length of the RSA public key is greater than or equal to 512 bits, than special secure mode is not required for SYG IM form.

TKE Workstation

The TKE workstation (Version 3 or later) uses the IBM 4758 card. The TKE workstation (Version 2) uses the IBM 4755 card. There are many changes to the TKE workstation and software. If you have a TKE workstation (Version 3 or higher) that connects to a host system running z/OS V1 R1 or lower, install APAR

OW46381 on the system(s) running releases prior to z/OS V1 R2 to ensure proper communications between the TKE workstation and ICSF. For detailed information on these changes, refer to *z/OS ICSF TKE Workstation User's Guide 2000*.

Migrating from OS/390 V2 R4 ICSF

The following sections describe activities and considerations that should be considered when migrating from V2 R4 ICSF. For a list of other migration activites, see "Common Migration Activities for z/OS ICSF, OS/390 ICSF and ICSF/MVS Version 2 Release 1" on page 52.

Installation Exits

The following differences in installation exits occurred between OS/390 V2 R4 ICSF and OS/390 V2 R5 ICSF and should be considered if upgrading from OS/390 V2 R4 ICSF or previous releases of ICSF. If you have already applied APAR OW31961 to OS/390 V2 R4 ICSF, you already have these changes.

- An additional parameter in the Single-record, Read-write installation exit identifies the accessed key data set as either the CKDS or the PKDS.
- The Key Generation Utility Program Exit Parameter Block (KGXP) contains a new subfield to hold the third key part for triple-length DATA keys.
- Any installation with either a Single-record, Read-write exit or a KGUP exit should recompile the exit.

Migrating from ICSF/MVS Version 2 Release 1

The following sections describe activities and considerations that should be considered when migrating from ICSF/MVS Version 2 Release 1. For a list of other migration activites, see "Common Migration Activities for z/OS ICSF, OS/390 ICSF and ICSF/MVS Version 2 Release 1" on page 52.

z/OS ICSF supports all versions of the cryptographic feature hardware. Customers who have OS/390 Enterprise Servers, OS/390 Multiprise servers or the IBM @server zSeries with the Cryptographic Coprocessor Feature can migrate to z/OS ICSF across their entire installation.

CKDS

A Version 1 Release 2 customer who shares a CKDS among multiple instances of ICSF need not migrate all instances of ICSF at the same time. Although, once new key types are added to the CKDS, the following considerations apply when sharing the CKDS:

- After you change a CKDS to contain the ANSI X9.17 enablement keys, all instances of ICSF/MVS Version 1 Release 2 that share that CKDS must have PTF UW90181 installed. This PTF was shipped against ICSF/MVS Version 1 Release 2 and was rolled up into Version 2 Release 1.
- You can share a CKDS that contains a limited authority importer key. However, OS/390 ICSF or ICSF/MVS 2.1 running on S/390 Enterprise Servers and S/390 Multiprise servers must perform any CKDS reencipherment.
- once new system keys, double-length MAC keys, IMP-PKA keys, etc. (introduced in ICSF/MVS 2.1) are added to the CKDS, it is sharable with instances of ICSF/MVS which do not support these keys. A CKDS reencipher operation must be performed on a system which supports these key types.

- once keys with non-CCF control vectors are added to the CKDS, a CKDS reencipher operation must be invoked from a system which has a PCI Cryptographic Coprocessor installed.
- once keys of type IMPORTER, EXPORTER, PINGEN, PINVER, IPINENC, or OPINENC which have non-CCF control vectors are added to the CKDS, a toleration APAR OW43926 must be installed on the non-OS/390 V2 R10 ICSF systems. The APAR ensures that ICSF services will fail a request to use a key which contains a non-CCF control vector.

Installation Exits

The following differences in installation exits occurred between OS/390 V2 R4 ICSF and OS/390 V2 R5 ICSF and should be considered if upgrading from OS/390 V2 R4 ICSF or previous releases of ICSF. If you have already applied APAR OW31961 to OS/390 V2 R4 ICSF, you already have these changes.

- An additional parameter in the Single-record, Read-write installation exit identifies the accessed key data set as either the CKDS or the PKDS.
- The Key Generation Utility Program Exit Parameter Block (KGXP) contains a new subfield to hold the third key part for triple-length DATA keys.
- Any installation with either a Single-record, Read-write exit or a KGUP exit should recompile the exit.

Migrating from ICSF/MVS Version 1

The following sections describe activities and considerations that should be considered when migrating from ICSF/MVS Version 1 Release 2 and Version 1 Release 1.

The following differences in installation exits occurred between OS/390 V2 R4 ICSF and OS/390 V2 R5 ICSF and should be considered if upgrading from ICSF/MVS Version 1.

- An additional parameter in the Single-record, Read-write installation exit identifies the accessed key data set as either the CKDS or the PKDS.
- The Key Generation Utility Program Exit Parameter Block (KGXP) contains a new subfield to hold the third key part for triple-length DATA keys.
- Any installation with either a Single-record, Read-write exit or a KGUP exit should recompile the exit.

Depending on which release of ICSF/MVS Version 1 you are migrating from, you will have different options to consider.

Migrating from ICSF/MVS Version 1 Release 2

You can use ICSF/MVS Version 1 Release 2 applications on ICSF without reassembling or relinking. This includes CUSP or PCF applications if you are running in compatibility mode.

If your installation is currently using ICSF/MVS Version 1 Release 2, consider the following:

- **CKDS**

  If you share a CKDS among multiple instances of ICSF/MVS you do not have to migrate all instances of ICSF/MVS at the same time. Although, once new key types are added to the CKDS, the following considerations apply when sharing the CKDS:

- After you change a CKDS to contain the ANSI X9.17 enablement keys, all instances of ICSF/MVS that share that CKDS must have PTF UW90181 installed. This PTF was shipped against ICSF/MVS Version 1 Release 2 and was rolled up into Version 2 Release 1.
- once a CKDS is modified to contain the ANSI X9.17 enablement keys, all instances of ICSF/MVS that share the CKDS must have APAR OW13633 installed.
- once new system keys (double-length MAC keys, IMP-PKA keys, etc) introduced in ICSF/MVS 2.1 are added to the CKDS, it is sharable with instances of ICSF/MVS which do not support these keys. A CKDS reencipher operation must be performed on a system which supports these key types.
- once keys with non-CCF control vectors are added to the CKDS, a CKDS reencipher operation must be invoked from a system which has a PCI Cryptographic Coprocessor installed.
- once keys of type IMPORTER, EXPORTER, PINGEN, PINVER, IPINENC, or OPINENC which has non-CCF control vectors are added to the CKDS, a toleration APAR OW43926 must be installed on the non-OS/390 V2 R10 ICSF or non-z/OS systems. The APAR ensures that ICSF services will fail a request to use a key which contains a non-CCF control vector.

- **ICSF/MVS Version 1 Release 2 Cryptographic Key Data Set**

  An ICSF/MVS Version 1 Release 2 CKDS will work on ICSF with no changes.

- **Installation exits**

  The following differences in installation exits occurred between OS/390 V2 R4 ICSF and OS/390 V2 R5 ICSF and should be considered if upgrading from ICSF/MVS Version 1 Release 2.

  - An additional parameter in the Single-record, Read-write installation exit identifies the accessed key data set as either the CKDS or the PKDS.
  - The Key Generation Utility Program Exit Parameter Block (KGXP) contains a new subfield to hold the third key part for triple-length DATA keys.
  - Any installation with either a Single-record, Read-write exit or a KGUP exit should recompile the exit.

  If you have user exits for ICSF/MVS Version 1 Release 2, they will work with ICSF with no modification.

  Since the RACF Security Access Facility (SAF) interface fully supports ICSF, OS/390 ICSF does not include the security exit routines that were provided with the previous release of ICSF. If you are using these security exit routines with ICSF/MVS Version 1 Release 2, you will need to migrate to the full SAF/RACF support. This support is available as a part of the Security Server option.

- **Master Key Entry**

  With ICSF on S/390 G3 Enterprise Server, or higher and the S/390 Multiprise, there are several options for master key entry. The option that is right for your application depends on the security requirements of your installation. The options include:

  - Pass Phrase Initialization allows the casual user to enter a pass phrase on the ICSF panels to set both DES and PKA master keys and initialize the CKDS. The value of the master keys is a repeatable function of the pass phrase. For this reason, the security of the pass phrase is critical to the security of the system.
  - The Clear Master Key Entry process allows the user to enter master key parts directly into the Cryptographic Coprocessor Feature through the use of ISCF

panels. In this procedure, the key parts appear briefly in the clear in host storage within the address space of the TSO user who is entering the keys and within the ICSF address space. When the master keys are stored in the secure Cryptographic Coprocessor Feature, these address spaces are cleared.

– The optional Trusted Key Entry (TKE) workstation (feature code 0806 or 809) replaces the physically secure hardware master key entry path available on bipolar processors with a logically secure channel implemented through an APPC (TKE Version 2) attachment. Installations that require this level of security need the TKE workstation, which comes fully configured by IBM Customized Solutions.

Migrating from ICSF/MVS Version 1 Release 1

You can use ICSF/MVS Version 1 Release 1 applications on ICSF without reassembling or relinking. This includes CUSP or PCF applications if you are running in compatibility mode.

Note: One exception to this is that ICSF/MVS Version 1 Release 1 key labels with a non-zero qualifier need to be RENAMED to an ICSF supported key label. Because the new label differs from the ICSF/MVS Version 1 Release 1 label, the applications need to be changed.

Note that once you create a CKDS that contains the ANSI enablement keys, all instances of ICSF/MVS that share that CKDS must have PTF UW90181 installed. This PTF was shipped against Version 1 Release 2 and was rolled up into Version 2.

If your installation is currently using ICSF/MVS Version 1 Release 1 and you are migrating to ICSF, consider the following:

- **ICSF/MVS Version 1 Release 1 Cryptographic Key Data Set**

  ICSF provides a conversion program to migrate an ICSF/MVS Version 1 Release 1 CKDS to an ICSF compatible CKDS. You can run the CKDS conversion program from either ICSF/MVS Version 1 Release 1 or OS/390 ICSF.

- **Installation exits**

  If you have user exits for ICSF/MVS Version 1 Release 1, they will work with OS/390 ICSF or z/OS with no modification.

- **Key labels**

  ICSF/MVS Version 1 Release 1 supports a key label of up to 8 bytes and multiple key types per label in the CKDS. ICSF/MVS Version 1 Release 2 introduced support for an extended key label of up to 64 bytes and required unique key labels for data-encrypting, data-translating, MAC-generating, and MAC-verifying keys. ICSF continues to support the 64-byte key label. The ICSF CKDS and KGUP will continue to support multiple key types per label for importer and exporter key-encrypting keys and PIN keys under the following conditions. You must use either KGUP or the KEU to enter the keys, and the key label cannot conflict with other unique label restrictions.

  RACF and security exit key protection in ICSF/MVS Version 1 Release 2 are by *label* rather than *label.type*. You need to rewrite any current RACF or security exit profiles that are based on *label.type*.

Converting a Version 1 Release 1 CKDS to z/OS ICSF Format

z/OS ICSF provides a conversion program, CSFCVR1, that converts a Version 1 Release 1 CKDS to the z/OS ICSF format. You can run the conversion program from either Version 1 Release 1 or from z/OS ICSF if you ensure that the system meets the following conditions:

- If you are running the CSFCVR1 conversion program on a Version 1 Release 1 system, you must fully initialize the Version 1 Release 1 system to the point of enabling application services.

  The job control language for CSFCVR1 **must** STEPLIB to the entire z/OS ICSF load library. Running the conversion program in Version 1 Release 1 does not involve loading a new master key or initializing a new CKDS (as in z/OS ICSF). For these reasons, it is the recommended conversion option.
- If you are running the CSFCVR1 conversion program on a z/OS ICSF system, you must fully initialize the z/OS ICSF system to the point of enabling application services.

  This means that you must first load a new master key and then complete the initialization of a z/OS ICSF CKDS. ICSF uses this new CKDS during the conversion process, but it is not the target of the conversion.

The next two sections describe how the conversion program runs and how to start it.

How the Conversion Program Works

The conversion program remaps each record in the Version 1 Release 1 CKDS to the new format. With the exception of the label and qualifier fields, the conversion program copies these records identically. During the conversion, ICSF calculates a new authentication code for each converted CKDS record. The conversion program does not call any exits and does not support the use of an override file.

Converting to the New Label Format

The conversion program remaps the 8-byte Version 1 Release 1 label field to the 64-character label field, padded on the right with blanks. If the input CKDS record contains nonzero information in the key qualifier field, the program converts the entire 8-byte hexadecimal field to a 16-byte EBCDIC character field. It appends 16-byte EBCDIC character field to the right of the new key label preceded by a single period (.).

For example, suppose you have a Version 1 Release 1 input record that contains a label of METOYOU, a type of EXPORTER, and a qualifier of X'1102920000000000'. This is converted to a ICSF label of METOYOU.1102920000000000, padded on the right with blanks to the full 64-byte label field.

Existing applications that use the old 8-byte label and count on the CKDS Retrieval Exit to select a key based on a nonzero qualifier do not work with the OS/390 ICSF or z/OS CKDS. You can use the KGUP RENAME verb to rename these changed labels to a valid 64-byte label.

Running the Conversion Program

You run the conversion program by submitting a batch job. On the EXEC statement, specify PGM=CSFCVR1.

The following example is the job control language that runs the conversion program:

```
//DAFRANK3 JOB
//CONVERT  EXEC PGM=CSFCVR1
//CSFVSRC  DD   DSN=ICSFR1.HCRP100.CKDS,DISP=SHR
//CSFVNEW  DD   DSN=ICSFV2.HCRP210.CKDS,DISP=SHR
//CSFVRPT  DD   SYSOUT=*
//
```

All the data sets necessary to run the conversion program are specified using DD statements.

The conversion program uses the following data sets:

CSFVSRC
: The ICSF/MVS Version 1 Release 1 CKDS containing entries that you want to convert into the z/OS ICSF format and place in the output CKDS. This is the source CKDS for the conversion.

CSFVNEW
: An empty disk copy of an ICSF CKDS. This is the z/OS ICSF CKDS into which the conversion program places key entries. The data set must be defined and empty before you run the conversion program.

CSFVRPT
: The activity report that the conversion program creates. The report contains a summary of the conversion process and includes error messages. The report lists only changed labels by their converted OS/390 ICSF label. The report provides the following counts:
  - The total labels processed, including system labels
  - The labels processed, where the qualifier was not binary zeros and was appended to the existing label

Attention: If a conversion program run ends prematurely, the results of the job are unpredictable. You should not read a CKDS involved in the conversion into storage for use. For a description of the conversion program return codes, see the explanation of message CSFV0026 in *z/OS ICSF Messages*.

When you run the conversion program, the program produces information about the conversion in an activity report. The activity report lists each record with a changed label and any error messages. The activity report also lists the data sets that were used in the conversion and a summary of processing. The summary of processing contains totals of the number of records that were processed and the number of labels that were changed.

Figure 4 on page 67 is an example of an activity report with two changed label conversions.

```
CRYPTOGRAPHIC CONVERSION ACTIVITY REPORT              DATE: 2001/06/01 (YYYY/MM/DD) TIME: 10:13:09 PAGE: 1
>>>CSFV0402 DAVE001.1993111200000000 PINGEN CREATED FROM A RELEASE 1 RECORD WITH A NON-ZERO QUALIFIER FIELD.
>>>CSFV0402 EXPN0E.1993111200000000 EXPORTER CREATED FROM A RELEASE 1 RECORD WITH A NON-ZERO QUALIFIER FIELD.
>>>CSFV0012 CONVERSION PROCESSING COMPLETED.  RETURN CODE = 0.

  CKDS DDNAME      Dataset Name
  ------------     -------------
  CSFVSRC          ICSFR1.HCRP100.CKDS
  CSFVNEW          ICSFO4.HCRP210.CKDS

Total number of CKDS record conversions processed    =        153
Total number of modified label conversions processed =          2
```

Figure 4. Example of a Version 1 Release 1 to ICSF z/OS Conversion Activity Report

In this example, ICSF converted 153 CKDS records. Two of the Version 1 Release 1 records contain nonzero qualifier fields, so the conversion program changed the labels for these records. The report shows the new labels. In this conversion, a Version 1 Release 1 record with a label of DAVE001 and a type of PINGEN contained the nonzero qualifier field X'1993111200000000'. The resulting OS/390 ICSF label and type for this record are DAVE001.1993111200000000 PINGEN. Similarly, the Version 1 Release 1 EXPORTER key with the label EXPN0E and a qualifier field of X'1993111200000000' was converted to EXPN0E.1993111200000000. You can use the KGUP RENAME verb to rename these changed labels to a valid 64-byte label.

After listing the changed key labels, the activity report lists the data sets the conversion program used in the conversion. ICSFR1.HCRP100.CKDS is the Version 1 Release 1 CKDS the program converted. ICSFO4.HCRP210.CKDS is the output z/OS ICSF CKDS where the conversion program placed the converted entries.

The activity report ends with the conversion processing completed message and a return code.

Migrating from 4753-HSP

ICSF provides key management callable services that are identical to the 4753-HSP verbs of the same name. Key management applications that are developed for the 4753-HSP and use these common verbs can be run on OS/390 ICSF or z/OS ICSF without reassembly. You will, however, need to relink them.

If your installation is currently using the 4753-HSP and you are migrating to OS/390 ICSF or z/OS ICSF, consider the following:

- **4753-HSP cryptographic key storage**

  Internal key tokens for ICSF and the 4753-HSP are not interchangeable. Key token migration for the 4753 exists through the optional TKE Version 3 Workstation. TKE Version 3 supplies a 4753 Migration Utility. It allows you to migrate internal DES key tokens from the 4753 to ICSF. Key exchange between the two systems is through the external key token. To migrate keys from the 4753-HSP to ICSF, you must first establish an exporter/importer key relationship between the 4753-HSP and ICSF. You can then write an application to export keys from the 4753-HSP key storage and import them into the ICSF CKDS. You can perform this type of key exchange only with CCA-defined keys, which have the same control vectors on key-encrypting keys. If your 4753-HSP installation includes non-CCA key types in key storage, you need to generate a special exporter/importer key-encrypting key pair on the 4753-HSP. The exporter

key-encrypting key nullifies the CV value that is used on the 4753-HSP, and the importer key-encrypting key includes the CV value that is needed at ICSF.

- **Callable Services**

  If you are migrating to OS/390 V2 R10 ICSF, the following differences should be considered. For more information on individual callable services, refer to *z/OS ICSF Application Programmer's Guide*, SA22-7522.

  – Clear Key Import - This service produces an internal DATA token with a control vector usable on the Cryptographic Coprocessor Feature. If a valid internal token is supplied as input to the service in the target *key_identifier* field, that token's control vector will not be used in the encryption of the clear key value.

  – Control Vector Generate - supports a subset of the TSS control vector key-usage keywords.

  – Control Vector Translate - if the *kek_key_identifier* parameter is specified as a label, and the identified token has a key type of IMPORTER or EXPORTER, then the label must be unique in the CKDS.

  – Clear PIN Generate Alternate, Clear PIN Generate and Encrypted PIN Verify do not provide support for the GBP-PINO calculation method when these services are routed to the PCI Cryptographic Coprocessor for execution.

  – Data Key Import - Data Key Import does not support direct write of the target token to the ICSF CKDS.

  – Key Generate - will not support the following *key_type_1* and *key_type_2* combinations for any *key_form*.

```
CIPHER      CIPHERXI
CIPHER      CIPHERXL
CIPHER      CIPHERXO
DECIPHER    CIPHERXO
ENCIPHER    CIPHERXI
CIPHERXI    CIPHER
CIPHERXL    CIPHER
CIPHERXO    CIPHER
CIPHERXO    DECIPHER
CIPHERXI    ENCIPHER
CIPHERXL    CIPHERXL
CIPHERXI    CIPHERXO
CIPHERXO    CIPHERXI
DATAXLAT    Null-CV
```

    In addition, the following *key_type_1* and *key_type_2* combinations which are supported by 4753 have slightly different support with OS/390 V2 R10 ICSF. These pairs will be supported only for the OPEX, EXEX, and IMEX key forms, and the only allowable control vectors will be those supported by the Cryptographic Coprocessor Feature.

```
DATA        DATAXLAT
DATAXLAT    DATAXLAT
```

    Key Generate does not support direct write of the target token to the ICSF CKDS.

  – Key Import - Key Import does not support direct write of the target token to the ICSF CKDS.

  – Key Token Build - Key types ADATA, AMAC, CIPHERXI, CIPHERXL, CIPHERXO, UKPTBASE are not supported. Rule array keywords KEY-REF, ADAPTER, READER, CARD, ACTIVE, INACTIVE, CLEAR-IV, NO-IV, CBC, X9.23, IPS, CUSP, X9.9-1, MACLEN4, MACLEN6, and MACLEN8 are not supported. The *master_key_verification_number* parameter has been replaced

by the *master_key_version_number* parameter. The *master_key_version_number* parameter is examined only if the KEY keyword is specified, and in this case must be zero. If KEY and INTERNAL are both specified in the rule array, the service will check for the existence of the rule array keyword MKVP. If MKVP is specified, the service will make use of the last parameter specified. The *key_register_number*, *secure_token*, and *initialization_vector* parameters are ignored. The *pad_character* parameter must have a value of zero.

– Multiple Clear Key Import - This service produces an internal DATA token with a control vector usable on the Cryptographic Coprocessor Feature. If a valid internal token is supplied as input to the service in the target *key_identifier* field, that token's control vector will not be used in the encryption of the clear key value.
– PKA Key Token Build - RSA-OPT rule array keyword is not supported. Optimized (Chinese Remainder Theorem) RSA keys are built using the RSA-CRT rule array keyword.
– PIN services do not support the OEM-1 PIN block format.
– Prohibit Export service - does not support DATA, MAC, or MACVER keys which have standard control vectors (for example, control vectors supported by the Cryptographic Coprocessor Feature).

  Prohibit Export does not support direct write of the target token to the ICSF CKDS.
– Secure Key Import - does not adjust key parity or support double-length DATA keys. Use the Multiple Secure Key Import service to process double-length DATA keys.
– It is not possible to migrate use of CIPHER keys from the 4753. CIPHER key types are not supported on the Cryptographic Coprocessor Feature.
– CVARPINE key can be built, generated, imported, or exported. They cannot be used since the Encrypted PIN Generate Alternate service is not supported.
– The following CCA services are not supported by ICSF:
  - Key Record List
  - Key Token Change
  - Key Token Parse
  - Clear PIN Verify
  - Cryptographic Variable Decipher
  - Encrypted PIN Generate Alternate
– Support for the following key types is not supported by ICSF: ACIPHER, ADATA, AMAC, CIPHERXI, CIPHERXL, CIPHERXO, and UKPTBASE.
– Support for the UDF/UDP control vector bit is not supported by ICSF.
– Chinese Remainder Theorem optimized key tokens with private key section identifier of X'05' are not supported by OS/390 V2 R10 ICSF. ICSF supports CRT tokens with a private key section identifer of X'08'.
– If PBVC is specified in the format control parameter of the PIN profile for the Clear PIN Generate Alternate service, the PIN Translate service, or the Encrypted PIN Verify service, only control vectors and extraction methods valid for the Cryptographic Coprocessor Feature may be used.
– Key management services such as Key Generate, Key Import, Data Key Import, and Prohibit Export do not support direct write of the target key token to the ICSF CKDS.
– During initialization of a PCI Cryptographic Coprocessor, an Environment Identification, or EID, of zero will be set in the card. This will be interpreted by

the PKA Symmetric Key Import service to mean that environment identification checking is to be bypassed. Thus, it is possible for a key-encrypting key RSA-enciphered at a node (EID) to be imported at the same node.

- **Key labels**

  ICSF/MVS Version 1 Release 2 and above supports an extended key label of up to 64 bytes. Although the 4753-HSP also supports a 64-byte key label, there are additional key label formatting restrictions that do not apply to ICSF. The 4753-HSP key label consists of one to five name tokens that are separated by periods. Each name token includes one to eight alphanumeric or national string characters. ICSF, therefore, can accept all 4753-HSP key labels, but the 4753-HSP cannot accept all ICSF key labels. For more information on key label formatting restrictions, refer to *IBM Transaction Security System: Concepts and Programming Guide: Volume I, Access Controls and DES Cryptography*.

  ICSF/MVS Version 1 Release 2 and above, like the 4753-HSP, requires unique key labels for data-encrypting keys, data-translation keys, and MAC keys. To maintain compatibility with ICSF/MVS Version 1 Release 1, however, KGUP will continue to allow multiple key types per label for importer, exporter, and PIN keys under the following conditions. Use either KGUP or the KEU to enter the keys, and ensure that the key labels do not conflict with other unique label restrictions.

- **UDX (User Defined Extension) support**

  Beginning with OS/390 V2 R10 ICSF, ICSF support is provided for UDX capabilities. UDX routines are developed by special contract with IBM and are only distributed to authorized customers.

  The UDX function is invoked by a ″installation-defined″ or generic callable service. The callable service is defined in the Installation Options data set (UDX parameter) and the service stub is link-edited with the application. The application program calls the service stub which accesses the UDX installation-defined service. There is a one-to-one correspondence between a specific generic service in ICSF and a specific UDX command processor in the PCI Cryptographic Coprocessor. The administrator, through ISCF panels, performs UDX authorization processing on each PCI Cryptographic Coprocessor. Authorization is not LPAR specific. See *z/OS ICSF Administrator's Guide*, SA22-7521, for additional information on authorizing User Defined Extensions.

  Beginning in z/OS V1 R2, support for writing your own UDX has been added. See the *UDX Reference and Guide* and the *4758 Custom Software Developer's Toolkit Guide* for additional information. These, and other publications related to the IBM 4758 Coprocessor can be obtained in PDF format from the Library page located at http://www.ibm.com/security/cryptocards.

  See *z/OS ICSF System Programmer's Guide*, SA22-7520, for details on installation-defined callable services and a description of the UDX parameter in the installation options data set.

Chapter 5. Compatibility and Coexistence of 4753-HSP and ICSF

The Transaction Security System products provide a range of cryptographic facilities. These facilities can be implemented throughout an organization using a compatible set of services at both workstation and host locations. One component of the Transaction Security System is the channel-attached IBM 4753 Network Security Processor (NSP) and its supporting software. The 4753 NSP can be installed at the IBM System/370, IBM System/OS/390 MVS or OS/390 host locations. The IBM Network Security Processor Support Program (referred to as 4753-HSP) provides host software support for the 4753 NSP. The Network Security Processor Control Program runs in the IBM 4753 NSP and processes encryption requests that are received from the host. If your installation is currently using 4753-HSP, you can either add ICSF and the Cryptographic Coprocessor Feature to your OS/390 host (where it can coexist with 4753-HSP) or migrate to ICSF. For more information about the Transaction Security System products, refer to *IBM Transaction Security System: General Information Manual and Planning Guide*.

Because both 4753-HSP and ICSF support the CCA, applications developed to run with 4753-HSP may run with ICSF without recompiling if they contain common verbs.

This chapter gives a brief overview of 4753-HSP and ICSF coexistence considerations. See “Migrating from 4753-HSP” on page 67 for migration considerations.

Running 4753-HSP and ICSF on the Same z/OS System

Although the 4753-HSP and ICSF can coexist in the same z/OS environment on a logical partition of a S/390 or z/OS complex, some restrictions apply.

Both systems can run simultaneously in noncompatibility mode. However, because both systems support the CCA API, they use the same verbs to call cryptographic services. For this reason, you must link your applications with the appropriate library routines to ensure that they are routed to the correct system. Use 4753-HSP stubs to link applications that are intended for 4753-HSP. Use ICSF stubs in SYS1.SCSFMOD0 to link applications that are intended for ICSF.

Both ICSF and 4753-HSP are capable of running CUSP/PCF compatibility mode, but only one system can provide this service at a time. Use of compatibility mode is effectively serialized by ownership of the CVTCCVT field.

The two systems use the external key token for key exchange. Internal key tokens are not interchangeable between 4753-HSP and ICSF, and each system uses different control vectors internally for data keys.

Generally, 4753-HSP provides additional function for a given service call. Be sure to use the common subset of services when an application operates with both of the systems. (Concurrent use of 4753-HSP and ICSF is beyond the scope of this book.)

z/OS ICSF supports a PKA implementation that differs from the Transaction Security System PKA implementation. The Transaction Security System supports both PKA92 and PKA96 versions. Applications that are written to one PKA version will not run on the other PKA version. Because ICSF does not support PKA92,

4753-HSP techniques that use RSA keys that have been implemented in PKA92 for DES key distribution are incompatible with ICSF applications. For PKA96, APIs are the same for services that ICSF and the 4753-HSP have in common. With the addition of RSA key support in the PKA Generate function, it becomes easier for the PKA96 to move to ICSF. The main difference is that the 4753-HSP does not support the Digital Signature Standard (DSS). RSA digital signatures that use ISO9796 formatting can be exchanged between the two products.

Chapter 6. Installation-Defined Callable Services

This chapter contains Programming Interface information.

See Appendix E, "Running HCR7708 on a IBM @server zSeries 990", on page 195 if you're running in this environment.

ICSF provides callable services that perform cryptographic functions. For example, the ICSF encipher callable service enciphers data. You call and pass parameters to a callable service from an application program. See *z/OS ICSF Application Programmer's Guide* for a description of the ICSF callable services.

Besides the callable services that ICSF provides, you can write your own callable services; these are known as *installation-defined callable services*.

Attention: Only an experienced system programmer should attempt to write an installation-defined callable service. The writing and installation of such a service require a thorough knowledge of system programming in an z/OS environment. If, without having this knowledge, you attempt to write or to install installation-defined callable services, you run the risk of seriously degrading the performance of your system and causing complete system failure.

To write an installation-defined callable service, you must first write the callable service and link-edit it into a load module. Then define the service in the installation options data set. Use the SERVICE installation option keyword to specify a number to identify the service and the load module that contains the service.

You must also write a service stub. To run an installation-defined callable service, you call a service stub from your application program. The service stub connects the application program with the installation-defined callable service. In the service stub, you specify the service number that identifies the callable service.

During ICSF startup, ICSF loads the load module that contains the service into the ICSF address space with the ICSF callable services. ICSF binds the service with the service number that you specified in the installation options data set.

This chapter describes how to perform the following tasks:
- Write a callable service.
- Define a callable service.
- Write a service stub.

Writing a Callable Service

An installation-defined callable service receives parameters from the application program when the program calls the service stub that is associated with the service. An installation-defined service can also access information in the secondary parameter block (SPB). The address of the SPB is passed in register 0. See "The Secondary Parameter Block" on page 103 for a description of the SPB.

The service receives control with the following characteristics.
- Supervisor state
- Key 0
- APF authorized
- TCB or SRB mode
- Cross memory mode

- AR mode
- AMODE(31)
- RMODE(ANY)

The service can change the characteristics during their processing. However, the service must return to its caller with the same characteristics as on entry.

You must write the services in assembler, because you are in Access Register and cross memory mode, and the addresses of some of the parameters you may access are ALET-qualified. In particular, parameters passed into a callable service are in the user's address space, which you can access with an ALET of 1. See *z/OS MVS Programming: Extended Addressability Guide* for information about cross memory and AR mode.

Contents of Registers

The contents of the registers on entry to the callable service are:

Register 0 Address of the secondary parameter block (SPB)
Register 1 Address of the parameter list
Register 2–13 Unpredictable
Register 14 Return address
Register 15 Service entry point address

The contents of the registers on exit from the callable service are:

Register 0 Reason code
Register 1–14 Same as on entry
Register 15 Return code

Figure 5 on page 75 shows an example of entry and exit code for a generic service.

```
MYSERV    CSECT
MYSERV    AMODE  31
MYSERV    RMODE  ANY
          USING  *,15
          B      PROLOG                Branch around header text
          DC     C'some text'
          DC     C'compile date/time'
PROLOG    EQU    *
          DROP   15
          BSM    R14,0
          BAKR   14,0                  Save callers info on stack
          LAE    12,0                  Clear access register 12
          LR     12,15                 Load reg 15 into 12
PROGSTRT  EQU    *
          USING  MYSERV,12             Set up base register
*                                       addressability
          .
          .
          .
          Get dynamic area for program
          .. STORAGE OBTAIN or CELLPOOL or own scheme ...
          .
          .
          Free dynamic area for program
          .
          .
          .
RETURN    L      0,REASON_CODE         Put reason code in reg 0
          L      15,RETURN_CODE        Put return code in reg 15
          PR
```

Figure 5. Example of a Service Entry and Exit

The example uses the instructions BAKR and PR to replace standard linkage. With these instructions, you no longer need to pass the save area in a register.

If the callable service ends abnormally, ICSF takes a system dump. The ICSF service functional recovery routine (FRR) PROTECTS an installation-defined service. You can, however, write your own recovery routine.

Checking the Parameters

For the ICSF-defined services, ICSF checks the integrity of user-passed parameters. An error in a parameter that causes a system abend does not cause a system dump. For an installation-defined callable service, you must perform your own integrity checking of parameters. An error in a user parameter that results in a system abend causes a system dump. You can suppress the system dump by setting a bit on in the SPB. To suppress the dump, set the bit on before you check the integrity of the parameters. This bit (the SPBTERM bit) is the third bit of the flag byte at offset 16 in the SPB.

Link-Editing the Callable Service

After you write the callable service, you need to link-edit it into a load module, and install the load module into an APF authorized library. ICSF uses the following normal search order to locate the service:

- Job pack area
- Steplib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

Defining a Callable Service

Use the SERVICE keyword in the installation options data set to specify information about the callable service. ICSF uses this information at ICSF startup to enable the service. See “Create the Installation Options Data Set” on page 14 for more information about ICSF installation options.

The SERVICE keyword has the following syntax:

SERVICE(*service-number***,***load-module-name***,FAIL(***fail-option***))**

The service-number is a number that identifies the service to ICSF. The valid service numbers are 1 through 32767, inclusive. The load-module-name is the name of the module that contains the service your installation wrote. During ICSF startup, ICSF loads the module and binds it to the service number you specified.

Using the fail-option, you specify the action ICSF takes if the loading of the service ends abnormally. ICSF loads all installation-defined services at ICSF startup.

Specify one of the following values for the fail-option:

YES ICSF abends if your service cannot be loaded.

NO ICSF continues to start if your service cannot be loaded.

If the callable service ends abnormally while it is processing, ICSF does not end.

The following SERVICE installation option statement identifies a specific installation-defined service to ICSF:

```
SERVICE(50,KSUST,FAIL(NO))
```

When ICSF starts, it binds the service number 50 to the load module KSUST, which contains the callable service you wrote. Because the fail option is NO, if your service cannot be loaded, ICSF continues to start anyway.

Writing a Service Stub

Besides writing the callable service itself, you must write a service stub, which is the connection between the application program and the installation-defined service. In an application program, you call the service stub, which accesses the installation-defined service. The service stub can be any name you choose to call it.

The service stub must do the following:

- Check that ICSF is active.
- Place the service number for the installation-defined callable service into register 0.
- Call the IBM-supplied processing routine, CSFAPRPC.

CSFAPRPC is used to access the callable services on ICSF. In the service stub, you must call CSFAPRPC. ICSF stores the address of the CSFAPRPC entry point in the CCVTPRPC field of the ICSF cryptographic communication vector table (CCVT). After you call CSFAPRPD, the system calls the callable service that corresponds to the service number in register 0. “The Cryptographic Communication Vector Table (CCVT)” on page 167 describes the format of the CCVT.

The contents of the registers on entry to the service stub are:

Register 0 Unpredictable

Register 1 Address of the parameter list
Register 2–13 Unpredictable
Register 14 Return address
Register 15 Service stub entry point address

The contents of the registers on exit from the service stub are:
Register 0 Reason code
Register 1–14 Same as on entry
Register 15 Return code

To run an installation-defined callable service, an application program calls the service stub. You must link-edit the service stub with the application program that calls the service stub. Any application program that calls a service stub must be link-edited with the service stub.

To call an installation-defined service from an application program, use the following statement:

```
CALL  <service-stub-name> <service-parameters>
```

The service-stub-name is the name of the service stub for the installation-defined callable service. The service-parameters are the parameters you want to pass to the installation-defined service. You supply the parameters according to the syntax of the programming language that you use to write the application program.

Figure 6 shows an example of a service stub for an installation-defined callable service.

```
**** START OF SPECIFICATIONS *****************************************
*                                                                     *
*    MODULE NAME = CSFGEN                                             *
*    DESCRIPTIVE NAME = SERVICE STUB                                  *
*                                                                     *
*    FUNCTION =                                                       *
*    THIS IS A SAMPLE SERVICE STUB. IT IS MEANT TO BE LINKEDITED      *
*    WITH THE APPLICATION AND ENTERED VIA A CALL CSFGEN. THIS STUB    *
*    CAUSES THE EXECUTION OF THE SERVICE WITH SERVICE NUMBER = 50     *
*    (DECIMAL).                                                       *
*    MODULE TYPE = ASSEMBLER                                          *
*       PROCESSOR = ASSEMBLER                                         *
*       MODULE SIZE = ONE BASE REGISTER                               *
*                                                                     *
**** END OF SPECIFICATIONS *******************************************
 CSFGEN  START 0
 GENSNUM EQU   50
 CSFGEN  CSECT
 CSFGEN  AMODE 31
 CSFGEN  RMODE ANY
 MAINENT DS    0H
         USING *,R15
         LAE   R15,0(R15,0)
         L     R15,=A(CICSTEST)
         BAKR  0,R15              PR from CICSTEST will restore GPRs
         LTR   R15,R15
         BC    2,NOCICS
 *
```

Figure 6. Example of a Service Stub (Part 1 of 5)

```
 YESCICS DS    0H
         SAC   0
         STM   R14,R12,12(R13)
         LR    R12,R15
         DROP  R15
         USING MAINENT,R12
         LR    R3,R0
         B     NORMAL
       *
         NOCICS  DS    0H
         USING MAINENT,R12
         BSM   R14,0
         BAKR  R14,0
         LAE   R12,0
         LR    R12,R15
         SLR   R13,R13
 ************************************************************
 * At this point, R0 must contain the service number.
 *                If we are to call the TRUE, R13 is non-zero
 *                R1 points to the caller's parameter list.
 ************************************************************
 NORMAL   DS    0H
          LA    R0,GENSNUM           R0 gets service number
          SLR   R10_ZERO,R10_ZERO
          LR    RC,R10_ZERO
          L     R2,CVTPTR
          USING CVT,R2
          L     R2,CVTABEND
         CLR   R2,R10_ZERO
         BC    8,NOICSF
         USING SCVTSECT,R2
         L     R2,SCVTCCVT
         CLR   R2,R10_ZERO
         BC    8,NOICSF
         USING CCVT,R2
         TM    CCVTSFG1,B'00110000' IS ICSF ACTIVE
         BC    1,YESICSF
NOICSF   LA    RC,12                Set return code to 12 decimal
         L     R7,RETURN_CODE_PTR(,R1)
         ST    RC,RETURN_CODE(,R7)
         SLR   R0,R0
         L     R7,REASON_CODE_PTR(,R1)
         ST    R0,REASON_CODE(,R7)
         B     FINISHED
YESICSF  DS    0H
*********************************************************************
* Note that, if we're in CICS, the prolog code pointed R3 at the AFCB
* and R13 at the caller's savearea--they're still pointing.  Also, R0
* contains the service number, with the high order bit ON if the TRUE
 * has been tried and found wanting. In this last case, CSFAPRPD will
 * check the high order bit and not attempt to call the TRUE.
 * If R13 is zero, we're using the linkage stack. That means we can
 * call CSFAPRPC.
 * If R13 is not zero, we're using non-stack linkage. That means the
 * caller's savearea will be used. CSFAPRPD uses this kind of linkage.
 * But note that CSFAPRPD won't return here. Instead, it will return
 * directly to the caller--that is, to the owner of the only save
 * area around.
 *********************************************************************
          CLR   R13,R10_ZERO
          BC    8,EXECPRPC
          L     R15,CCVTPRPD
          BALR  R14,R15
```

Figure 6. Example of a Service Stub (Part 2 of 5)

```
         LR    RC,R15
         B     FINISHED
EXECPRPC L     R15,CCVTPRPC
         BALR  R14,R15
         LR    RC,R15
FINISHED DS    0H
*
*****************************************************************
* This routine uses the linkage stack to save the caller's regs
* if this is not a CICS environment.  In CICS, it uses the save
* area pointed to by register 13.  So the epilog code takes one
* of two forms.  If this is CICS (i.e. if R13 is non-zero),
* return is via LM and BR 14.  If this is not CICS, return is
* via PR.
*
* On return, the PR of ESA linkage does not restore registers
* 0, 1, 14 and 15.  In the LM of normal BR 14 linkage, however,
* everything but 13 gets restored.  Since this routine has no
* autodata, there's no way to pass back return and reason codes
* unless we leave 0 and 15 intact.  The solution is to deviate
* slightly from normal BR 14 linkage and restore only registers
* 1 through 12 and 14.
*****************************************************************
          LTR   R13,R13
          BC    8,ENDNOCICS
ENDCICS   LR    R15,RC
          L     R14,SAVE14(,R13)
          LM    R1,R12,24(R13)
          BR    R14
*
ENDNOCICS DS    0H
          LR    R15,RC
          PR
************************************************************************
************************************************************************
**  CICSTEST:  Decides whether this is a CICS environment
************************************************************************
************************************************************************
CICSTEST DS   0H
        LAE   R12,0               Clear AR 12
        LR    R12,R15             Addressability via R12
        USING CICSTEST,R12
        L     R15,=A(CSFGEN)       R15 gets caller's base reg
        L     R2,CVTPTR            GET CVT POINTER
        USING CVT,R2
        L     R2,CVTABEND          AND SECONDARY CVT POINTER
        USING SCVTSECT,R2
        L     R2,SCVTCCVT          POINT TO CSF CCVT
        LTR   R2,R2                IS CRYPTO INSTALLED?
        BZ    RETRN                IF NOT, GO HOME
        USING CCVT,R2
        TM    CCVTSFG1,B'00110000' IS ICSF ACTIVE
        BNO   RETRN                IF NOT , GO HOME
* Check for wait list routine
        TM    CCVTCICS,B'10000000' Q. CCVTPRPA ON?
        BZ    RETRN                     no---No CICS capability
        TM    CCVTCICS,B'01000000' Q. CCVTCKWL ON?
        BZ    CKWLHERE                  no---use imbedded routine
 *                                      yes--use installed routine
```

Figure 6. Example of a Service Stub (Part 3 of 5)

```
         LA    R0,GENSNUM           R0 gets service number
         LR    R3,R1                R3 saves R1
         LR    R4,R14               R4 saves R14
         LR    R5,R15               R5 saves R15
         L     R15,CCVTCKWL         R15 gets routine address
         BALR  R14,R15              Go check for CICS
         LR    R0,R15               Save return code in R0
         LR    R15,R5               Restore R15
         LR    R14,R4               Restore R14
         LR    R1,R3                Restore R1
         LTR   R0,R0                Q. CICS?
         BZ    RETRN                  no---return
*                                     yes--pass info along
         O     R15,M_CICS           Enable high bit of R15 to CICS
         B     RETRN                Return
* Cannot use installed routine.  Use imbedded routine
CKWLHERE DS    0H                   Imbedded check for TRUE routine
         SLR   R0,R0                Init R0 to 0
         CPYA  R8,R12               Zero AR 8
         SLR   R8,R8                Init R8 to 0
         USING PSA,R8
         L     R8,PSATOLD           R8->TCB
         USING TCB,R8
         LTR   R8,R8                Q.  Is there a TCB?
         BC    8,RETRN                no---return
*                                     yes--check state and key
         CPYA  R11,R12              Zero AR 11
         LA    R11,1                Get PSW state and key in R6
         ESTA  R6,R11
         LR    R7,R6                Copy of state & key in R7
         N     R7,M_KEY             Q.  problem key?
         BZ    RETRN                  no---return
*                                     yes--check state
         N     R6,M_STATE           Q.  problem state?
         BZ    RETRN                  no---return
*                                     yes--get the CICS eye-catcher
         LA    R6,2                 Set ARs 6 and 8 to home
         SAR   R6,R6
         SAR   R8,R6
         L     R8,TCBEXT2           R8->TCB extension
         USING TCBXTNT2,R8
         ICM   R4,B'1111',TCBCAUF   R4 gets AFCX address
*                                   Q.  Address there?
         BZ    RETRN                      no---return
*                                         yes--check eye-catch
         CLC   0(4,R4),CICS_EYE     Q.  CICS?
         BNE   RETRN                  no---return
*                                     yes--pass info along
         LR    R0,R4               R0 gets the AFCX pointer
         O     R15,M_CICS           Enable high order bit of R15
RETRN    DS    0H
         DROP  R12                   Free R12
          PR                         Return from CICSTEST subroutine
*
          LTORG
          DS    0D
```

Figure 6. Example of a Service Stub (Part 4 of 5)

```
*
GENSDATA DS    0F
R10_ZERO EQU   10
RC       EQU   05
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
INPUT_PARMS     EQU 0,8,C'C'
RETURN_CODE_PTR EQU INPUT_PARMS,4,C'A'
REASON_CODE_PTR EQU INPUT_PARMS+4,4,C'A'
RETURN_CODE     EQU 0,4,C'F'
REASON_CODE     EQU 0,4,C'F'
*
SAVAREA  EQU   0,72,C'C'
SAVE14   EQU   SAVAREA+12,4,C'A'
SAVE01   EQU   SAVAREA+24,4,C'A'
SCVTSPTR EQU   CVTABEND,4,C'F'
TCBPTR   EQU   PSATOLD,4,C'F'
         DS    0D
*
         DS    0F                   Align
M_KEY    DC    X'00800000'         Problem key mask
M_STATE  DC    X'00010000'         Problem state mask
M_NOCICS DC    X'7FFFFFFF'         Not-CICS mask
M_CICS   DC    X'80000000'         Yes-CICS mask
         DS    0D
CICS_EYE DC    CL4'AFCX'           CICS eye catcher
*
         IHAPSA
         TITLE 'DSECT CVT'
         CVT DSECT=YES
         TITLE 'DSECT SCVT'
         IHASCVT DSECT=YES
         TITLE 'DSECT TCB'
         IKJTCB
         TITLE 'DSECT CCVT'
         CSFCCVT
*
         END
```

Figure 6. Example of a Service Stub (Part 5 of 5)

In Figure 6 on page 77, the service stub, CSFGEN, checks that ICSF is active, places the service number 50 into register 0, and calls CSFAPRPC.

The service number 50 must be bound to the installation-defined service by using the SERVICE keyword in the installation options data set. The service number is bound to the service when ICSF interprets the SERVICE installation option

statement and loads the service at ICSF startup. To run the callable service that is associated with service number 50, call the service stub CSFGEN from an application program.

For flexibility, to create a service stub for a different installation-defined callable service, you can copy an existing service stub and just change the service number that you load into register 0.

Chapter 7. Installation Exits

See Appendix E, “Running HCR7708 on a IBM @server zSeries 990”, on page 195 if you’re running in this environment.

Your installation can define exit routines to supplement the Integrated Cryptographic Service Facility (ICSF), the key generator utility program (KGUP), and the PCF conversion program. Exit routines are programs that programmers at your installation write to allow you to “customize” an application. Your installation may need to perform specific functions with the data that your cryptographic application manipulates. At various points in processing, ICSF, KGUP, and the PCF conversion program release control to an exit routine.

Some common uses for installation exits include:
- Identifying and verifying users
- Accessing alternate data sets
- Manipulating input commands
- Manipulating output data

This chapter describes the various types of exit points in ICSF and the functions that your exits can perform.

Attention: Only an experienced system programmer should use the ICSF installation exits. Writing an exit routine and installing a new exit are tasks that require a thorough knowledge of system programming in an OS/390 and z/OS environment. An unknowledgeable programmer who attempts to write exit routines or to install new exit points, runs the risk of seriously degrading the performance of your system and causing complete system failure.

Types of Exits

ICSF provides several types of exit points:
- Exits that are called during initialization, stopping, and modification of ICSF itself, which are known as the mainline exits
- Exits that are called from the callable services
- An exit called from the PCF conversion program
- An exit called when you update the CKDS with a key that is entered through the key entry hardware or during conversion program processing
- An exit called when records are retrieved from the in-storage CKDS
- Security exits that are called during initialization and stopping of ICSF, during a call to a callable service, and when accessing a CKDS entry
- An exit called at various points during KGUP processing

The following sections briefly describe the different types of exits available in ICSF.

Note: Although IBM no longer supplies security exit routines, the exit points still remain.

Mainline Exits

You can supply three exits that are called during ICSF initialization. You can also define an exit routine to run after an operator issues the STOP command and another exit to run after the MODIFY command. Thus, mainline exits can run at the following five different points:

- Initialization points
  - Before ICSF initialization
  - After ICSF reads and interprets the installation options
  - Before the completion of ICSF initialization
- When an operator issues a STOP ICSF command
- When an operator issues a MODIFY ICSF command

You can use a mainline exit to alter values in the Cryptographic Communication Vector Table, to end ICSF, or to change ICSF installation options. For more information about the mainline exits, see “Mainline Installation Exits” on page 88.

Exits for the Callable Services

Each of the callable services in ICSF calls an exit before and after processing. *z/OS ICSF Application Programmer's Guide* describes the callable services in greater detail.

You can use a callable service exit to change, augment, or replace processing or to bypass the IBM-supplied processing for the service entirely. “Callable Services Installation Exits” on page 95 gives further details about exits for the callable services.

The PCF CKDS Conversion Program Exit

The PCF conversion program changes a CKDS from PCF to ICSF CKDS format. See Chapter 3, “Migration from PCF to z/OS ICSF”, on page 33 for more information about the conversion program.

ICSF provides three exit points for the same exit routine:

- During the initialization of the conversion program
- While the conversion program is processing individual records
- During the ending of the conversion program

See “PCF Conversion Program Installation Exit” on page 108 for more information about the conversion program installation exit (CSFCONVX).

The Single-record, Read-write Exit

Certain ICSF processes read records from or write records to the CKDS. These processes include running a conversion program, refreshing and reenciphering the CKDS, and using the key entry hardware to enter a key. When these processes read or write CKDS records, they call the exit. You can customize the processing of a CKDS record read-write with the single-record, read-write exit (CSFSRRW). See “Single-record, Read-write Installation Exit” on page 111 for more information about the single-record, read-write exit.

When application programs write records to or read records from the PKDS, ICSF calls the single-record, read-write exit.

The Cryptographic Key Data Set Entry Retrieval Exit

You can use certain callable services to manage keys on ICSF. A callable service can access a key in the in-storage CKDS by specifying a key label. For more information about the callable services, see *z/OS ICSF Application Programmer's Guide*.

When a callable service requests a record from the in-storage CKDS by label, ICSF calls the CKDS entry retrieval exit. For instance, you can use this exit to perform a specific search of the installation data field in the record. See "Cryptographic Key Data Set Entry Retrieval Installation Exit" on page 105 for more information about the CKDS entry retrieval exit.

Security Exits

You can supply four different exits to control access to resources on ICSF. ICSF calls the security exits at the following points:

- During CSF initialization
- During CSF termination
- When an application calls an ICSF callable service
- When an entry in the in-storage CKDS is accessed

See "Security Installation Exits" on page 114 for more information about the security exits.

The KGUP Exit

You use KGUP to generate and maintain keys in the CKDS. KGUP creates key values that systems can use in key exchanges. The ICSF administrator uses job control language to start KGUP,and specifies information to KGUP through the use of a control statement.

As opposed to the five different mainline exits, ICSF provides one exit for KGUP processing that is called at four different points. ICSF calls the KGUP exits at the following points:

- During KGUP initialization
- Before KGUP processes a key that is identified by a control statement
- Before KGUP updates the CKDS
- During KGUP termination

The KGUP exit receives a parameter that identifies the exit's calling point. Thus, the installation exit can perform different functions at each of the calls.

You can use the KGUP exit to change key values, make a copy of a CKDS entry, or end KGUP. "Key Generator Utility Program Installation Exit" on page 118 gives a more detailed description of the KGUP exit.

Entry and Return Specifications

All of the exits described in "Types of Exits" on page 83 use standard linkage conventions on entry and return from the exits.

Registers at Entry

The mainline exits have the following register contents on entry:

Register	Contents
0	Address of the exit parameter block (EXPB)

1	Address of a parameter list
2–12	Not applicable
13	Address of register save area
14	Return address
15	Entry point address

The callable service exits have the following register contents on entry:

Register	Contents
0	Address of the exit parameter block (EXPB)
1	Address of a parameter list
2–13	Not applicable
14	Return address
15	Entry point address

The CKDS entry retrieval installation exit has the following register contents on entry:

Register	Contents
0	Not applicable
1	Address of a parameter list
2–12	Not applicable
13	Address of register save area
14	Return address
15	Entry point address

The conversion program, single-record, read-write, and KGUP exits have the following register contents on entry:

Register	Contents
0	Not applicable
1	Address of a control block (CVXP, RWXP, or KGXP, depending on the exit)
2–12	Not applicable
13	Address of register save area
14	Return address
15	Entry point address

The particular control blocks that are passed through register 0 or register 1 are described with each exit.

Registers at Return

Registers for all exits must contain the original contents on entry with the exception of register 15 which must contain a valid return code. See each exit for a list of valid return codes. The registers should contain the following information on return.

Register	Contents

0–14	Same as entry contents
15	Valid return code

Exits Environment

ICSF calls different types of exits in distinct environments. The exits differ regarding the mode in which they run and how they address data.

Mainline Exits

ICSF mainline exits run in task mode in the ICSF address space. All the passed storage pointers specify addresses in the ICSF address space and are not ALET qualified. There are essentially no restrictions on the use of z/OS services for these exits.

Callable Service Exits

ICSF calls the callable service exits in cross memory mode after a space switch PC. The exits run in the ICSF address space, which is the primary address space. The exits need to address parameters in the caller's address space, which is the secondary address space. In general, user-passed parameters, including the parameter list itself, are in the secondary address space. An exit that is running in access register (AR) mode using an ALET of 1 can access these parameters. For information about cross memory mode and AR mode, see *z/OS MVS Programming: Extended Addressability Guide*.

CKDS Entry Retrieval Exit

The exit runs in cross memory mode. The addresses of the CKDS records that are used by the exit are ALET-qualified. The exit receives both the current CKDS record address and the record's associated ALET as parameters in the exit parameter list. The exit must run in AR mode, and must use the information passed in the exit parameter list to access CKDS entries. For information about cross memory mode and AR mode, see *z/OS MVS Programming: Extended Addressability Guide*.

KGUP, Conversion Programs, and Single-record, Read-write Exits

The exits run in task mode in the caller's home address space. The exits do not run in cross memory mode and are not passed ALET-qualified storage pointers. There are essentially no restrictions on the use of z/OS services for these exits.

Security Exits

The initialization and termination security exits run in task mode in the ICSF address space. The passed storage pointers specify an address in the ICSF address space and are not ALET-qualified. There are essentially no restrictions on the use of z/OS services for these exits.

ICSF calls the security service exit and the security keys exit in cross memory mode after a space switch PC. The security service exit runs in the ICSF address space, which is the primary address space. The security key exit runs in cross memory and AR mode.

Exit Recovery

An ESTAE routine provides recovery for the mainline exits; the single-record, read-write exit; and the security initialization and termination exits. If an exit ends abnormally, the ESTAE routine intercepts the abnormal ending code and schedules a system dump. If the conversion program exit ends abnormally, the conversion program ends abnormally. If the KGUP exit ends abnormally, KGUP also ends abnormally. ESTAE routines provide recovery for the conversion program and KGUP.

The ICSF Functional Recovery Routine (FRR) provides recovery for the callable service exits, the CKDS entry retrieval exit, and the security service and key exits. If an exit ends abnormally, the FRR intercepts the abnormal ending code and schedules a system dump.

There are times during ICSF processing that ICSF suppresses dumps. For example, ICSF does not schedule dumps when integrity checking user data. This action avoids the possibility of user errors that can severely affect system performance. However, ICSF does write a record to SYS1.LOGREC if the error occurs.

When writing exits, you may also want to suppress dumps under certain circumstances. You can suppress dumps by setting a bit on in the SPB. This bit, the SPBTERM bit, is the third bit of the flag byte at offset 18 in the SPB. An exit might want to suppress dumps whenever the exit writes user storage. The exit can turn the bit on before the WRITE instruction and turn the bit off again after the instruction.

Mainline Installation Exits

ICSF begins when an operator issues a START command from the operator console. When ICSF issues this command, the initialization process begins.

After ICSF starts, operators can issue the MODIFY or STOP commands. You can define installation exits to customize ICSF at the initialization, stopping, and modification points.

Purpose and Use of the Exits

ICSF calls the mainline exits during the startup, modification, and shutdown stages. The exits allow your installation to change the initialization options, issue special messages, and bypass operator commands. Following is a description of each point at which ICSF calls mainline exit routines.

CSFEXIT1

ICSF calls this exit after an operator issues a START command, but before any processing takes place. You can use this exit to change the allocation of the installation options data set.

ICSF always calls the exit. If this exit does not exist, ICSF continues normal processing. If this exit exists, ICSF starts it.

CSFEXIT2

ICSF calls this exit during the initialization process after the installation options data set is read and interpreted. You can use this exit to change certain installation options.

CSFEXIT3
ICSF calls this exit just before ICSF initialization is complete. You can use this exit to issue commands to start other cryptographic work.

CSFEXIT4
ICSF calls this exit when an operator issues a STOP command. You can use this exit to decide to allow or disallow the STOP command.

CSFEXIT5
CSFEXIT5 receives the command input block (the string that is entered by the operator), so you can customize CSFEXIT5 to perform any processing you require. ICSF calls this exit when an operator issues a MODIFY command. ICSF provides the MODIFY command exit to allow each installation the flexibility of defining its own command. ICSF does no processing when an operator uses the MODIFY command. The MODIFY command is simply a call to CSFEXIT5.

Environment of the Exits

The exits receive control with the following characteristics:
- Supervisor state
- Key 0
- APF-authorized
- TCB mode
- Address Space Control mode=access register mode
- AMODE(31)
- RMODE(ANY)

The exits can change the characteristics during their processing. However, the exits must return to ICSF with the same characteristics as on entry.

Installing the Exits

Because ICSF calls CSFEXIT1 before any initialization occurs, the exit is not defined in the same way as the other exits. For all the mainline exits, install the load module that contains the exit into an APF-authorized library. ICSF uses the following normal OS/390 search order to locate the exit:
- Job pack area
- Steplib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

You must define CSFEXIT2, CSFEXIT3, CSFEXIT4, and CSFEXIT5 in the installation options data set. However, you *must not* define CSFEXIT1 in the installation options data set, and the load module name for the exit must be CSFEXIT1.

To define the exits in the installation options data set, define the ICSF exit point name and load module name on the EXIT keyword in the installation options data set. For information about the installation options data set, see “Changing Parameters in the Installation Options Data Set” on page 22. The EXIT keyword has the following syntax:

EXIT (*ICSF exit point name***,** *load module name***, FAIL (***options***))**

The **ICSF exit point name** portion of the keyword refers to the ICSF name for each exit, CSFEXIT2, CSFEXIT3, CSFEXIT4, and CSFEXIT5. The **load module name** is the name of the load module that contains the exit. The name can be any valid

name your installation chooses. The **FAIL** portion of the EXIT keyword specifies the action ICSF takes if the exit cannot be loaded. The valid FAIL options are:

NONE Initialization continues even if exits cannot be loaded.
SERVICE Initialization continues even if exits cannot be loaded.
EXIT Initialization continues even if exits cannot be loaded.
ICSF End ICSF if exits cannot be loaded.

You must specify a FAIL option. If you do not, ICSF returns an error message, abnormally ends, and generates an SVC dump when attempting to load the exit.

Input

All mainline exits receive the address of an exit parameter block (EXPB) passed in register 0. Each exit receives the address of an address list passed in register 1. Each address in the list points to a parameter.

Figure 7 illustrates the contents of register 0 and EXPB for the mainline exits.

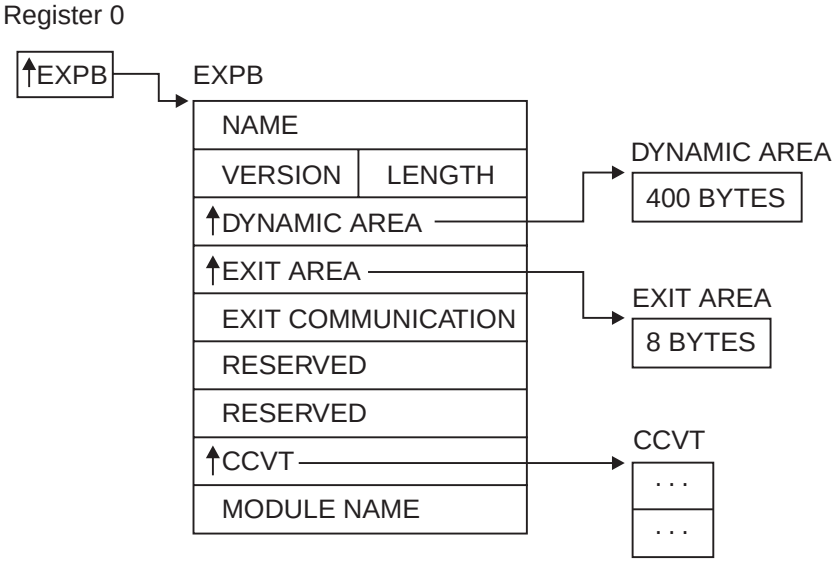


Figure 7. EXPB Control Block for Mainline Exits

Both the mainline exits and the callable services exits receive the address of EXPB in register 0. Some of the fields in EXPB are used only by the callable service exits and are reserved fields for the mainline exits.

The Exit Parameter Block

Table 3 describes the contents of the exit parameter block.

Table 3. EXPB Control Block Format for Mainline Exits

Offset (Dec)	Number of Bytes	Description
0	4	Name. The name of the control block. This field contains the character string `EXPB`.

Table 3. EXPB Control Block Format for Mainline Exits (continued)

Offset (Dec)	Number of Bytes	Description
4	2	Version. The version of the control block. This field contains the character string `01`.
6	2	Length. The length of the control block. The value of this field is 40 in decimal.
8	4	Dynamic area address. The address of a 400-byte area that the exit can use as a dynamic area.
12	4	Exit area address. The address of an 8-byte area the exits can use to communicate with each other. ICSF does not check or change this field.
16	4	Exit communication area. A character string that can be used for communication between the exits. The field is initialized to zero before CSFEXIT1 is called, and ICSF does not modify this field.
20	4	Flags. **Reserved**. The flag field is used only by the exits for the callable services. The field contains binary zeros for the mainline exits.
24	4	Secondary parameter block (SPB) address. **Reserved**. The SPB is used only by the exits for the callable services. The field contains binary zeros for the mainline exits.
28	4	CCVT address. Address of the Cryptographic Communication Vector Table (CCVT). “The Cryptographic Communication Vector Table (CCVT)” on page 167 describes the CCVT in greater detail.
32	8	Module name. The installation exit’s load module name. The field contains the value of the load module name you specified on the EXIT keyword in the installation options data set. The field is 8 bytes of characters, and the value is left-justified and padded with blanks.

Parameters

All mainline exits receive an address list that uses standard entry linkage. Register 1 points to the address list. Each address in the list points to a parameter. Tables in the next four sections describe the parameters for each of the mainline exits.

CSFEXIT1: The following table describes the parameters for CSFEXIT1:

Table 4. CSFEXIT1 Parameters

Parameter	Number of Bytes	Description
1	8	The data set name (DDNAME) of the installation options data set.
2	Variable	The command input block for the START command. The command control block is mapped by IEZCIB.

When ICSF calls this, the Cryptographic Communication Vector Table exists, but the table is not yet complete.

CSFEXIT2 and CSFEXIT3: Both CSFEXIT2 and CSFEXIT3 receive the same parameters. Table 5 describes these parameters.

Table 5. CSFEXIT2 and CSFEXIT3 Parameters

Parameter	Number of Bytes	Description
1	44	A character string that is the CKDS name specified in the CKDSN installation option.
2	4	A decimal value that is the maximum length permitted for data passed to callable services specified in the MAXLEN installation option. Beginning with z/OS V1 R2, the MAXLEN parameter may still be specified in the options data set, but only the maximum value limit will be enforced (2147483647). If a value greater than this is specified, an error will result and ICSF will not start.
3	4	ICSF environmental options. **Note:** Do not change bits 2, 4, and 5. Byte 1: **Bit** — **Meaning When Set On** **0** — Special secure mode allowed. **1** — Special secure mode enabled. **2** — Reserved and must be zero. **3** — Key authentication required. **4** — The hardware has gone from active to inactive. **5** — First start of ICSF during this IPL. **6** — Security Sever (RACF) checking required for authorized callers. **7** — PCF coexistence. Bytes 2–4: Reserved
4	4	Address of the exit name table. Table 7 on page 93 describes the exit name table.

CSFEXIT4 and CSFEXIT5: Both CSFEXIT4 and CSFEXIT5 receive the same parameters. Table 6 describes these parameters.

Table 6. CSFEXIT4 and CSFEXIT5 Parameters

Parameter	Number of Bytes	Description
1	44	A character string that is the CKDS name specified in the CKDSN installation option.
2	4	A decimal value that is the maximum length permitted for data passed to callable services specified in the MAXLEN installation option. Beginning with z/OS V1 R2, the MAXLEN parameter may still be specified in the options data set, but only the maximum value limit will be enforced (2147483647). If a value greater than this is specified, an error will result and ICSF will not start.
3	4	ICSF environmental options. **Note:** Do not change bits 2, 4, and 5. Byte 1: **Bit** — **Meaning When Set On** **0** — Special secure mode allowed. **1** — Special secure mode enabled. **2** — Reserved and must be zero. **3** — Key authentication required. **4** — The hardware has gone from active to inactive. **5** — First start of ICSF during this IPL. **6** — Security Server (RACF) checking required for authorized callers. **7** — PCF coexistence. Bytes 2–4: Reserved
4	4	Address of the exit name table. Table 7 describes the exit name table.
5	Variable	The command input block. You can use the IEZCIB mapping macro to map the control block.

The Exit Name Table: The exit name table contains a list of all of the exits and their load module names. Table 7 describes the format of the exit name table.

Table 7. Format of the Exit Name Table

Offset (Dec)	Number of Bytes	Description
0	4	Exit name table ID. The value is always the character string `ENT`.
4	2	Exit name table version. The value is always the character string `01`.
6	2	Length of the exit name table. This value is in decimal.

Table 7. Format of the Exit Name Table (continued)

Offset (Dec)	Number of Bytes	Description
8	4	Number of entries in the array which is the number of exits ICSF supplies. This value is in decimal.
12	4	Subpool that the exit name table is in.
16	4	**Reserved.**
20	4	**Reserved.**
24	4	**Reserved.**
28	4	**Reserved.**
32	8	ICSF exit name 1. This value is a character string.
40	8	Installation load module name 1. This value is a character string.
48	4	Flags. Flag bytes. Only the first two bytes are used; bytes 3 and 4 are reserved. Byte 1: **Bit** — **Meaning When Set On** **0** — Exit has been requested by the installation. **1** — Exit has been loaded. **2** — Exit is active. **3** — If exit fails, end ICSF. **4** — If exit fails, do not call the exit again. **5** — If exit fails, fail the service. **6** — If exit fails, do nothing. **7** — Exit has failed previously. Byte 2: **Bit** — **Meaning When Set On** **0** — The exit should be called. **1** — The exit is available to the installation. **2** — If the security exit fails, fail the service. **3–7** — **Reserved.**
52	4	Address of the exit.
56	4	**Reserved.**
60	4	**Reserved.**
64	8	ICSF exit name 2. This value is a character string.
72	8	Installation load module name 2. This value is a character string.
80	4	Flags. See offset +48 for flag byte definitions.
84	4	Address of the exit.

Table 7. Format of the Exit Name Table (continued)

Offset (Dec)	Number of Bytes	Description
88	4	**Reserved.**
92	4	**Reserved.**
⋮		⋮
x	8	ICSF exit name a.
x+8	8	Installation load module name a.
x+16	4	Flags. See offset +48 for flags.
x+20	4	Address of the exit.
x+24	4	**Reserved.**
x+28	4	**Reserved.**

Return Codes

All mainline exits can pass back a return code in register 15. CSFEXIT1, CSFEXIT2, and CSFEXIT3 support the following decimal return codes:

Return Code	Description
0	Proceed with initialization.
16	End ICSF.

CSFEXIT4 supports the following decimal return codes:

Return Code	Description
0	Proceed with the STOP command.
4	Do not allow the STOP command to proceed.

CSFEXIT5 supports the following decimal return codes:

Return Code	Description
0	Continue processing.
4	End ICSF.

Any return codes other than those listed cause ICSF to end abnormally.

Callable Services Installation Exits

ICSF provides callable services that you can use to perform various cryptographic functions. Examples of these functions include enciphering and deciphering data, generating and verifying message authentication codes, generating and verifying PINs, and dynamically updating the CKDS and PKDS. You can define an installation exit for each of the callable services to customize processing. For a detailed description of the callable services, see *z/OS ICSF Application Programmer's Guide*.

Use the following general format to request a callable service:

```
CALL CSNBxxx (
             return_code
            ,reason_code
            ,exit_data_length
            ,exit_data
            ,parameter_5
            ,parameter_6
              .
              .
              .
            ,parameter_N)
```

Table 8 on page 97 lists the ICSF exit names for each of the callable services. The parameters that the application passes to a callable service are known as the callable service parameter list, and the parameters vary from service to service. “Parameters” on page 105 describes the callable services parameter lists in more detail.

Purpose and Use of the Exits

Each of the callable services has an installation exit. Each installation exit for a callable service has two exit points:

- **The Preprocessing exit point.** This exit point occurs after an application program calls a callable service, but before the callable service starts processing. For example, you can use this exit point to check or change the parameters that the application passes on the call, or to end the call. You can also perform additional security checks.
- **The Postprocessing exit point.** This exit point occurs after the callable service has finished processing, but before the service returns control to the application program. For example, you can use this exit point to check and change the return code from the service or perform cleanup processing.

Environment of the Exits

The exits receive control with the following characteristics:

- Supervisor state
- Key 0
- APF-authorized
- TCB or SRB mode
- Cross memory mode
- AR mode
- AMODE(31)
- RMODE(ANY)

The exits can change the characteristics during their processing. However, the exits must return to their caller with the same characteristics as on entry.

You must write the exits in assembler, because you are in AR and cross memory mode and the addresses of some of the parameters you may access are ALET-qualified. In particular, parameters passed into a callable service are in the user’s address space which you can access with an ALET of 1. Additionally, the CKDS entries are ALET-qualified, and the ALET itself is supplied as a parameter to the CKDS retrieval exit.

For information about cross memory and AR mode, see *z/OS MVS Programming: Extended Addressability Guide*.

Installing the Exits

You install an exit for a callable service by installing the load module that contains the exit into an APF-authorized library. ICSF uses the following normal search order to locate the exit:

- Job pack area
- Steplib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

Define the ICSF name and the load module name as a value on the EXIT keyword in the installation options data set. For more information about the installation options data set, see “Changing Parameters in the Installation Options Data Set” on page 22. The EXIT keyword has the following syntax:

EXIT (*ICSF name*, *load module name*, **FAIL (***options***))**

The **ICSF name** portion of the keyword refers to the ICSF name for each callable service exit. Note that the ICSF name for each callable service exit is the same as its CALLable name. Table 8 lists the ICSF names for each of the callable service exits. Table 9 on page 99 lists the ICSF names for each of the compatibility service exits. The **load module name** is the name of the load module that contains the exit. The name can be any valid name that your installation chooses. The **FAIL** portion of the EXIT keyword specifies the action ICSF takes if the exit cannot be loaded or it ends abnormally. The valid FAIL options are:

NONE No action is taken. The exit can be called again and will end abnormally again.

EXIT The exit is no longer available to be called again.

SERVICE The service or program that called the exit is no longer available to be called again.

ICSF ICSF or the key generator utility program or the PCF conversion program is ended, depending on the exit.

You must specify a FAIL option. If you do not, ICSF returns an error message, ends abnormally, and generates an SVC dump when attempting to load the exit. If the exit ends abnormally, the service call fails regardless of the fail option you specified. Fail options apply only to subsequent requests for the service.

Note: In the following table, CSFPKSC (PKSC interface) and CSFPCI (PCI interface), are a part of the product-sensitive programming interface.

Table 8. Callable Services and Their ICSF Names

Callable Service	***ICSF Name***
ANSI X9.17 EDC Generate	**CSFAEGN**
ANSI X9.17 Key Export	**CSFAKEX**
ANSI X9.17 Key Import	**CSFAKIM**
ANSI X9.17 Key Translate	**CSFAKTR**
ANSI X9.17 Transport Key Partial Notarize	**CSFATKN**
Ciphertext Translate	**CSFCTT**
Ciphertext Translate (with ALET)	**CSFCTT1**
Clear Key Import	**CSFCKI**
Clear PIN Encrypt	**CSFCPE**
Clear PIN Generate	**CSFPGN**

Table 8. Callable Services and Their ICSF Names (continued)

Callable Service	***ICSF Name***
Clear PIN Generate Alternate	**CSFCPA**
Control Vector Translate	**CSFCVT**
Cryptographic Variable Encipher	**CSFCVE**
Data Key Export	**CSFDKX**
Data Key Import	**CSFDKM**
Decipher	**CSFDEC**
Decipher (with ALET)	**CSFDEC1**
Decode	**CSFDCO**
Digital Signature Generate	**CSFDSG**
Digital Signature Verify	**CSFDSV**
Diversified Key Generate	**CSFDKG**
Encipher	**CSFENC**
Encipher (with ALET)	**CSFENC1**
Encode	**CSFECO**
Encrypted PIN Generate	**CSFEPG**
Encrypted PIN Translate	**CSFPTR**
Encrypted PIN Verify	**CSFPVR**
Key Export	**CSFKEX**
Key Generate	**CSFKGN**
Key Import	**CSFKIM**
Key Part Import	**CSFKPI**
Key Record Create	**CSFKRC**
Key Record Delete	**CSFKRD**
Key Record Read	**CSFKRR**
Key Record Write	**CSFKRW**
Key Test	**CSFKYT**
Key Test Extended	**CSFKYTX**
Key Translate	**CSFKTR**
MAC Generate	**CSFMGN**
MAC Generate (with ALET)	**CSFMGN1**
MAC Verify	**CSFMVR**
MAC Verify (with ALET)	**CSFMVR1**
MDC Generate	**CSFMDG**
MDC Generate (with ALET)	**CSFMDG1**
Multiple Clear Key Import	**CSFCKM**
Multiple Secure Key Import	**CSFSKM**
One Way Hash Generate	**CSFOWH**
One Way Hash Generate (with ALET)	**CSFOWH1**
PCI Interface	**CSFPCI**
PKA Decrypt	**CSFPKD**

Table 8. Callable Services and Their ICSF Names (continued)

Callable Service	***ICSF Name***
PKA Encrypt	**CSFPKE**
PKA Key Generate	**CSFPKG**
PKA Key Import	**CSFPKI**
PKA Key Token Change	**CSFPKTC**
PKA Public Key Extract	**CSFPKX**
PKDS Record Create	**CSFPKRC**
PKDS Record Delete	**CSFPKRD**
PKDS Record Read	**CSFPKRR**
PKDS Record Write	**CSFPKRW**
PKSC Interface	**CSFPKSC**
Prohibit Export	**CSFPEX**
Prohibit Export Extended	**CSFPEXX**
Random Number Generate	**CSFRNG**
Retained Key Delete	**CSFRKD**
Retained Key List	**CSFRKL**
Secure Key Import	**CSFSKI**
Secure Messaging for Keys	**CSFSKY**
Secure Messaging for PINs	**CSFSPN**
SET Block Compose	**CSFSBC**
SET Block Decompose	**CSFSBD**
Symmetric Key Export	**CSFSYX**
Symmetric Key Generate	**CSFSYG**
Symmetric Key Import	**CSFSYI**
Transform CDMF Key	**CSFTCK**
User Derived Key	**CSFUDK**
VISA CVV Service Generate	**CSFCSG**
VISA CVV Service Verify	**CSFCSV**

Note: The alias for Common Programming Interface (CPI) callable services is CSNBxxx. The alias for the ANSI X9.17 key management callable services is CSNAxxx. The alias for the PKA callable services is CSNDxxx.

Table 9. Compatibility Services and Their ICSF Names

Compatibility Service	***ICSF Name***
Encipher under Master Key	**CSFEMK**
Generate a key	**CSFGKC**
Import a key	**CSFRTC**
Cipher/Decipher	**CSFEDC**

Input

The installation exit for each callable service gets the address of the exit parameter block (EXPB) in register 0. ICSF obtains and initializes an EXP for every service call. Figure 8 illustrates the contents of register 0, and Table 10 illustrates the EXPB for the callable service exits.

Register 1 contains the address of an address list. Each address in the list points to a parameter. “Parameters” on page 105 describes the callable service parameter list. The parameters the exit receives are the same parameters that are passed on the call to the callable service. For more information about the parameters for each callable service, see *z/OS ICSF Application Programmer’s Guide*.

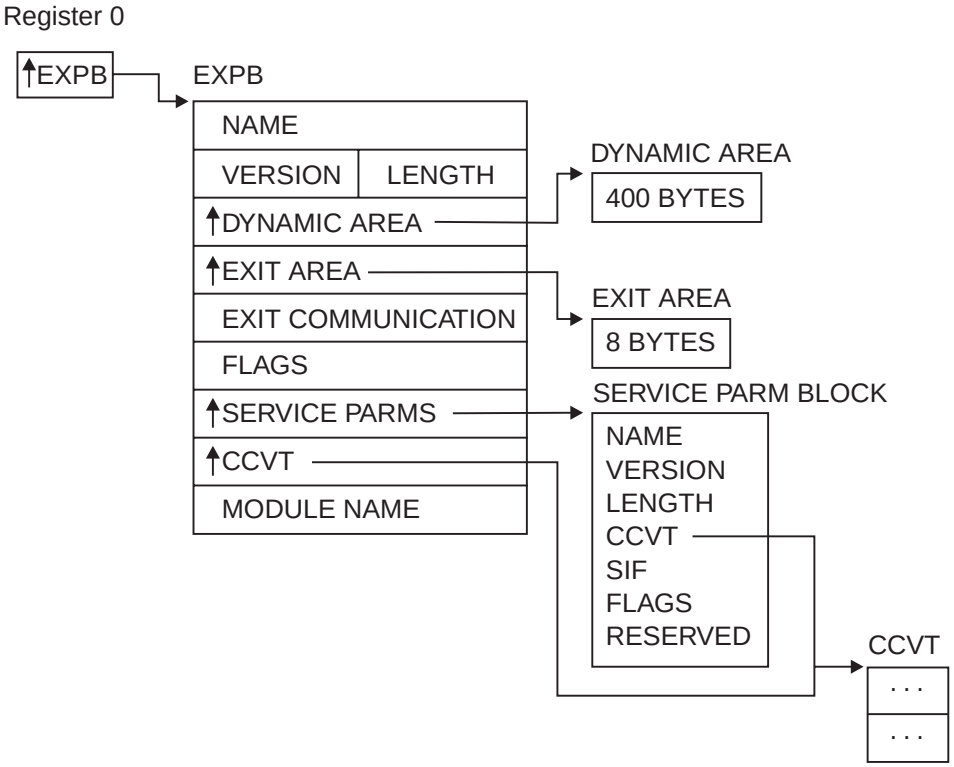


Figure 8. EXPB Control Block in the Callable Service Exits

The Exit Parameter Block

Table 10 describes the contents of the exit control block.

Table 10. EXPB Control Block Format for Callable Services

Offset (Dec)	Number of Bytes	Description
0	4	Name. The name of the control block. The field contains the character string `EXPB`.
4	2	Version. The version of the control block. The field contains the character string `01`.

Table 10. EXPB Control Block Format for Callable Services (continued)

Offset (Dec)	Number of Bytes	Description
6	2	Length. The length of the control block. The value is 40 in decimal.
8	4	Dynamic area. The address of a 400-byte area that the exit can use as a dynamic area.
12	4	Exit area address. The address of an 8-byte area for the preprocessing and postprocessing invocations of the exit to use for communication. ICSF does not check or change this field.
16	4	Exit communication area. A character string that can be used for communication between preprocessing and postprocessing invocations of a callable service exit.

Table 10. EXPB Control Block Format for Callable Services (continued)

Offset (Dec)	Number of Bytes	Description
20	4	Flags. A flag byte. Each bit setting (on/off) indicates a particular condition. ICSF sets bit 0 and an exit cannot change that bit. Your exit can set any of the other bits. **Bit** — **Meaning When Set On/Off** **0** — Postprocessing invocation./Preprocessing invocation. **1** — **Reserved.** **2** — Use the return and reason code that the exit places in register 0 and register 15 as the callable service's return code/reason code. Do not use the exit's return code as the service return code in registers 0 and 15. The exit can pass any valid return code in register 15 and any valid reason code in register 0. If this bit is set on, ICSF uses these codes as the callable service's return and reason codes. See "Return Codes" on page 105 for more information about using exit return codes. **3** — Do not call the postprocessing invocation of the callable service exit./Call the postprocessing invocation of the callable service exit. **4** — Bypass the callable service./Run the service. **5** — Use the return and reason code that the exit places in the service's parameter list./Do not store codes the exit places in the service's parameter list. The exit can pass any valid return and reason code in the first two parameters of the callable service's parameter list. "Parameters" on page 105 describes the callable service parameter list. **6** — CSFSKRC bypass input label parsing./CSFSKRC parse the input label. **7–31** — **Reserved.**
24	4	Secondary parameter block. The address of the secondary parameter block. The exit can use the SPB to determine the environmental information of the callable service. For a description of the SPB, see "The Secondary Parameter Block" on page 103.

Table 10. EXPB Control Block Format for Callable Services (continued)

Offset (Dec)	Number of Bytes	Description
28	4	CCVT. Address of the Cryptographic Control Vector Table (CCVT). For a description of the CCVT, see “The Cryptographic Communication Vector Table (CCVT)” on page 167.
32	8	Module name. The installation exit’s load module name. The field contains the value of the load module name you specified on the EXIT keyword in the installation options data set. The field is 8 bytes of characters, and the value is left-justified and padded with blanks.

The Secondary Parameter Block

Offset +24 of EXPB contains the address of the secondary parameter block (SPB). The exit can use the SPB to determine the environmental conditions of the callable service. Table 11 describes the contents of SPB.

Table 11. SPB Control Block Format

Offset (Dec)	Number of Bytes	Description
0	4	Name. The name of the control block. The field contains the character string `SPB`.
4	2	Version. The version of the control block. The field contains the character string `02`.
6	2	Length. The length of the control block. The value is 24 in decimal.
8	4	CCVT. The address of the Cryptographic Communication Vector Table (CCVT). For a description of the CCVT, see“The Cryptographic Communication Vector Table (CCVT)” on page 167.
12	4	Signal Information Word. Bytes 1–2 Reserved. Bytes 3–4 of the field contain the installation-assigned code number for an installation-defined callable service.

Table 11. SPB Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
16	4	Flags and Indicators. Each byte of this field is either an indicator byte or contains flag bits. The contents of each byte in the field are listed below. Byte 1—PSW key. This byte contains the original caller's program status word key. The first four bits are the key and the remaining four bits are zeros. Byte 2—Caller's state. Each bit in byte 2 indicates a condition of the caller's state. **Bit** — **Meaning When Set On** **0** — Caller was a PC entry from CSFASVC. You called SVC entry from a PCF compatibility macro. **1** — Original caller in AMODE(31). **2** — Original caller in AR mode. **3** — Original caller in SRB mode. **4** — Original caller in supervisor state or system key. **5–7** — **Reserved**. Byte 3—Flag byte 1. The first flag byte. Each bit that is set on indicates a particular condition. **Note:** These bits are informational. Do not change bits 0 and 1. **Bit** — **Meaning When Set On** **0** — The callable service is using a "storage access" ICSF instruction. **1** — ICSF local lock should be freed when recovery is entered. **2** — If recovery is entered, the recovery routine should take a system abend but not a system dump. **3** — An I/O subtask has been posted for a dynamic CKDS service call. **4** — I/O subtask posted for PKDS. **5** — NQAP in progress. **6–7** — **Reserved.** Byte 4—Flag byte 2 **Bit** — **Meaning When Set On** **0** — The callable service parameter list has a position for a return code. **1** — The callable service parameter list has a position for a reason code. **2** — Internal call to ICSF service. **3** — **SPBLATCH held for CCP array and not for CAMQ.** **4-7** — **Reserved.**
20	4	Protected storage pointer.
24	4	Protected storage length.
28	4	EDC buffer pointer.
32	4	EDC buffer length.

Table 11. SPB Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
36	4	Address of XPB.
40	8	ID for latch manager.
48	4	Address for ERPB.
52	4	Original caller's register 1.
56	4	Address of CPRB request storage.
60	4	Length of CPRB request storage.
64	4	Address of CPRB reply storage.
68	4	Length of CPRB reply storage.
72	4	CCPS address.
76	8	RTM token.
84	4	**Reserved.**

Parameters

Each callable service has a unique parameter list. Parameters 1–4 are always the return code, reason code, exit data length, and exit data. The other parameters differ with each service. The installation exit gets passed the address of the callable service parameter list in Register 1. For a description of each callable service's parameter list, refer to *z/OS ICSF Application Programmer's Guide*.

Return Codes

To use a return code and reason code that are set in the postprocessing exit, you must set bit 2 in Offset +20 of EXPB. Setting bit 2 on causes ICSF to return the return code from the exit in register 15 and the reason code in register 0. Even though the application program receives the codes from the exit in the registers, the program still receives the codes from the callable service in the parameter list. The return code is the first parameter, and the reason code is the second parameter in the list.

Some control languages can access registers more easily than others. For this reason, ICSF allows you to return the return code and the reason code in both the registers and the parameter list. To do this, set bit 5 as well as bit 2 in Offset +20 of EXPB. The application then receives the return code and the reason code from the exit in both the registers and the parameter list.

If you do not set either of or both of the flag bits, the callable service ignores any return or reason code from the exit. The application program receives the codes from the callable service in both the registers and the parameter list.

The exit can pass back any valid return code for each service. For a listing of each service's return codes, see *z/OS ICSF Application Programmer's Guide*.

Cryptographic Key Data Set Entry Retrieval Installation Exit

The cryptographic key data set entry retrieval installation exit (CSFCKDS) is called when a callable service requests an entry from the in-storage cryptographic key data set (CKDS) by label. ICSF calls this exit after it finds the record in the CKDS and before it returns the record to the callable service.

Purpose and Use of the Exit

The exit point lists the entry that matches a certain label and type. You can use the exit to check fields in a record and decide whether to use the record. The exit sets a return code that specifies whether to use the record or not. Use the *exit_data* parameter in the callable service to specify what the exit should use as a search value.

For example, you can use the CKDS entry retrieval exit to perform a specific search of the installation data field. An installation can specify whatever it chooses to in the installation data field. The exit can select a record that matches a certain key label and key type. You can check the record and accept or reject it based on the installation data field.

Environment of the Exit

The exit receives control with the following characteristics:

- Supervisor state
- Key 0
- APF-authorized
- TCB or SRB mode
- AR mode
- AMODE(31)
- RMODE(ANY)
- Cross memory mode

The exit can change the characteristics during its processing. However, the exit must return to its caller with the same characteristics as on entry.

The exit runs in the cross memory mode in the ICSF address space. The CKDS records are ALET-qualified. ICSF supplies the address and the ALET of a CKDS record as parameters to the CKDS retrieval exit.

For information about cross memory mode and AR mode, see *z/OS MVS Programming: Extended Addressability Guide*.

Installing the Exit

Install the CKDS entry retrieval exit by installing the load module that contains the exit into an APF-authorized library. ICSF uses the following normal z/OS search order to locate the exit:

- Job pack area
- Steplib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

Define the ICSF name and the load module name on the EXIT keyword in the installation options data set. “Changing Parameters in the Installation Options Data Set” on page 22 describes the installation options data set in further detail. The EXIT keyword has the following syntax:

EXIT (*ICSF name*, *load module name*, **FAIL (***options***))**

The **ICSF name** portion of the keyword refers to the ICSF name for the exit. The ICSF name for the CKDS entry retrieval exit is CSFCKDS. The **load module name** is the name of the load module that contains the exit. The name can be any valid

name that your installation chooses. The **FAIL** portion of the EXIT keyword specifies the action ICSF takes if the exit cannot be loaded or if it ends abnormally. The valid FAIL options are:

NONE Do not take any action.

EXIT Do not call this exit again. The exit will not receive control during subsequent attempts at CKDS retrieval.

SERVICE Fail the service. All subsequent attempts at CKDS entry retrieval fail.

ICSF End ICSF.

You must specify a FAIL option. If you do not, ICSF returns an error message, ends abnormally, and generates an SVC dump when attempting to load the exit. If the exit ends abnormally, the attempt at CKDS entry retrieval fails, regardless of the FAIL option you specified. FAIL options only apply to subsequent attempts at CKDS entry retrieval.

Input

The CKDS entry retrieval exit receives the address of an address list passed in register 1. Each address in the list points to a parameter. The address list exists in the ICSF address space, and register 1 is not ALET-qualified.

Table 12 describes the parameters for the CKDS entry retrieval exit.

Table 12. The CKDS Entry Retrieval Exit Parameters

Parameter	Description
1	The address of the current CKDS record. See Table 20 on page 150 for a description of the CKDS record format.
2	The address of the ALET of the current CKDS record. This record is a fullword address.
3	The address of the record that matches a certain label and type. This value is a fullword integer. The parameter is in the ICSF address space and the exit can access the parameter using an ALET of 0.
4	The address of the record chosen. This value is a fullword integer. The parameter is in the ICSF address space and the exit can access the parameter using an ALET of 0.
5	The address of the exit data length. This value is a fullword integer. The parameter is in the caller's address space, which is the secondary address space, and the exit can access the parameter using an ALET of 1.
6	The address of the exit data. For a description of exit data, see *z/OS ICSF Application Programmer's Guide*. The parameter is in the caller's address space, which is the secondary address space, and the exit can access the parameter using an ALET of 1.
7	The address of the secondary parameter block. See "The Secondary Parameter Block" on page 103 for a description of the secondary parameter block. The parameter is in the ICSF address space and the exit can access the parameter using an ALET of 0.

Return Codes

You can pass a return code back in register 15.

The valid decimal return codes are:

Return Code	Description
0	Use the record.
4	Do not use the record.

If you specify not to use any of the records that match the search value, ICSF returns control to the application. It returns with return code 12 and reason code 10024, which indicate that the exit rejected all the keys in the search.

PCF Conversion Program Installation Exit

Use the PCF conversion program to convert a CKDS from the Programmed Cryptographic Facility (PCF) format to the ICSF format. The conversion program converts each record in the PCF CKDS to the CKDS format that ICSF uses, and then writes the new record to an ICSF CKDS. The conversion program extends the label field to 64 bytes.

An ICSF CKDS record contains an installation data field that you can use to further identify the record. This field can contain any information about a record that your installation would like to use. You can use the conversion program exit to change the information in this field. You can also use the conversion program exit to have the conversion program not place a converted CKDS entry in the ICSF CKDS.

Chapter 3, “Migration from PCF to z/OS ICSF”, on page 33 contains more information about the PCF conversion program.

Note: The ICSF/MVS Version 1 Release 1 conversion program cannot run this exit.

Purpose and Use of the Exit

The PCF conversion program installation exit (CSFCONVX) is called at three points during processing of the conversion program:

- **During conversion program initialization.** This is known as the conversion preprocessing invocation. At this point, you can use the exit to change the ICSF CKDS header record installation data field.
- **During conversion program individual record processing.** This is known as the record processing invocation. At this point, the conversion program is converting the PCF entry but has not yet placed the entry into the ICSF CKDS. You can use the exit to change the installation data field in the entry for the ICSF CKDS. You can also specify that the conversion program not place the entry into the ICSF CKDS.
- **Just prior to conversion program termination.** This is known as the conversion postprocessing invocation. At this point, like the preprocessing exit point, you can use the exit to change the ICSF CKDS header record installation data field.

Environment of the Exit

The exit receives control with the following characteristics:

- Problem program state.
- APF-authorized
- TCB mode

- Address Space Control mode=primary
- AMODE(31)
- RMODE(ANY)

The exit can change the characteristics during its processing. However, the exit must return to its caller with the same characteristics as on entry.

The exit runs in task mode in the caller's own address space.

Installing the Exit

Install the load module that contains the exit into an APF-authorized library. ICSF uses the following normal OS/390 search order to locate the exit:

- Job pack area
- Steplib (if one exists)
- Joblib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

Define the ICSF name and load module name on the EXIT keyword in the installation options data set. For more information about the installation options data set, see "Changing Parameters in the Installation Options Data Set" on page 22. The EXIT keyword has the following syntax:

EXIT (*ICSF name*, *load module name*, **FAIL (***options***))**

The **ICSF name** portion of the keyword refers to the ICSF name for the exit. The ICSF name for the conversion program exit is CSFCONVX. The **load module name** is the name of the load module that contains the exit. This name can be any valid name that your installation chooses. The **FAIL** portion of the EXIT keyword specifies the action ICSF takes if the exit cannot be loaded. The valid FAIL options are **NONE**, **EXIT**, **SERVICE**, and **CSF**. For the conversion program exit, you can use the following options only:

NONE Initialization continues even if exit cannot be loaded.
ICSF Initialization ends if exit cannot be loaded.

You must specify a FAIL option. If you do not, ICSF returns an error message, ends abnormally, and generates an SVC dump when attempting to load the exit.

If the exit ends abnormally, the conversion program does also.

Input

ICSF supplies the address of the conversion program exit parameter block (CVXP) in register 1 each times it calls the PCF conversion program exit. The exit does not receive a parameter list. "Entry and Return Specifications" on page 85 gives a complete list of the registers on entry to the conversion program exit.

Table 13 describes the contents of the exit control block.

Table 13. CVXP Control Block Format

Offset (Dec)	Number of Bytes	Description
0	4	Name. The name of the control block. The field contains the character string `CVXP`.

Table 13. CVXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
4	2	Version. The version of the control block. The field contains the character string `01`.
6	2	Length. The length of the control block. The value is 28 in decimal.
8	4	Return Code. The value the exit returns. Valid decimal values for this field are: **Return Code** — **Description** **0** — Normal. **4** — Do not process the entry. **8** — End conversion program.
12	4	Address of the ICSF CKDS installation data area.
16	4	The value in decimal of the length of the ICSF CKDS installation data area.
20	1	Action. Bit 0 is set on if the action was to change an entry on the ICSF CKDS. Bit 0 is set off if the action was to add an entry to the ICSF CKDS. The rest of the bits in this byte are reserved.
21	1	Call Point. Indicates the invocation point of the exit. The exit cannot change this field and the conversion program does not use this field on return from the exit. You can determine the invocation point by the bit that is set on. **Bit** — **Meaning When Set On** **0** — Conversion preprocessing invocation. **1** — Conversion postprocessing invocation. **2** — Record processing invocation. **3-7** — **Reserved**.
22	6	Reserved.

Return Codes

You can pass a return code back to the conversion program in the CVXP control block (offset +8) or in register 15. The exit can use return codes to reject records for conversion processing or end the conversion program.

Return Code	Description
0	Normal.
4	Do not process the entry.
8	End conversion program.

Single-record, Read-write Installation Exit

ICSF provides an exit that is called when a record is read from or written to a CKDS or PKDS. ICSF calls the single-record, read-write (CSFSRRW) exit under the following conditions:

- The PCF conversion program converts a record into ICSF CKDS format. The conversion program calls the exit right before it writes a converted record to the ICSF CKDS.
- ICSF reenciphers a disk copy of a CKDS under a new master key. ICSF calls the exit two times during this processing; after ICSF reads a record to reencipher it and before ICSF writes the reenciphered record.
- ICSF refreshes the in-storage copy of a CKDS. ICSF calls this exit after reading a record from the disk copy to place into storage.
- You enter a key into a disk copy of the CKDS by using the key entry hardware. ICSF calls the exit two times during this processing: after reading a partial key from the CKDS, and before writing a record into a CKDS.
- An application creates, reads, writes, or deletes a record from the PKDS.

Using the exit, you can do such things as prevent the record from being processed, or add user information to the record.

Purpose and Use of the Exit

The exit receives a parameter block that describes the CKDS or PKDS record and the action occurring to the record. By setting a return code in the parameter block, the exit may affect the processing of the record. Depending on the return code, one of the following actions occurs:

- ICSF continues to read the record.
- ICSF does not read or write the record.
- ICSF does not read or write the entire CKDS or PKDS.

The parameter block contains the address of the CKDS or PKDS record. The exit can add information into the installation data field of the record. For integrity reasons, ICSF receives only changes to this particular field. If the exit sets a return code to continue processing, ICSF processes the record with this information.

The KGUP exit, the PCF conversion program exit, and the single-record, read-write exit can add information to the installation data field of the CKDS or PKDS header record to identify the data set. If the header record installation data field contains information identifying the CKDS or PKDS, the single-record, read-write exit can check the field to ensure that it is processing the correct data set. If the exit finds that it is processing the wrong CKDS or PKDS, the exit can set a return code to stop the processing of the entire data set.

You can use the exit to prevent processing of a record. You can check certain fields in the record and specify that the record not be processed. For example, during postprocessing conversion, you can prevent the processing of any record of a certain key type. However, the exit should never prevent processing of a record containing a system key because ICSF uses these keys in its processing. You differentiate a system key record from other key records by its key label. A system key record label contains all binary zeros. All other key labels contain an alphabetic first character with the remaining characters as either alphabetic or numeric.

Environment of the Exit

The exit receives control with the following characteristics:

- Problem program state
- APF-authorized
- TCB mode
- Address Space Control mode=primary
- AMODE(31)
- RMODE(ANY)

The exit can change the characteristics during its processing. However, the exit must return to ICSF with the same characteristics as on entry.

When the single-record, read-write exit is called, the exit parameter block is in the caller's address space. The exit is loaded in the caller's address space. The caller is either the PCF conversion program, the utility program (CSFEUTIL), or an ICSF panel.

Installing the Exit

Install the load module that contains the exit into an APF-authorized library. ICSF uses the following search order to locate the exit:

- Job pack area
- Steplib (if one exists)
- Joblib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

Define the ICSF name and load module name on the EXIT keyword of the installation options data set. For more information about the installation options data set, see "Changing Parameters in the Installation Options Data Set" on page 22. The EXIT keyword has the following syntax:

EXIT (*ICSF name*, *load module name*, **FAIL (***options***))**

The **ICSF name** portion of the keyword refers to the ICSF name for the exit. The ICSF name for the single-record, read-write exit is CSFSRRW. The **load module name** is the name of the load module that contains the exit. The name can be any valid name that your installation chooses. The **FAIL** portion of the EXIT keyword specifies the action ICSF takes if the exit cannot be loaded or ends abnormally. The valid FAIL options are:

NONE	Do not take any action.
EXIT	Do not call this exit again.
SERVICE	Fail the service that called the exit.
ICSF	Fail the service that called the exit.

You must specify a FAIL option. If you do not, ICSF returns an error message, ends abnormally, and generates an SVC dump when attempting to load the exit. If you specify FAIL(ICSF) and the exit cannot be loaded, ICSF initialization does not continue. If you specify FAIL(ICSF) and the exit ends abnormally, ICSF issues an advisory message that ICSF should be ended.

Input

The single-record, read-write exit receives the address of the address list passed in register 1. The first address in the address list is for the read-write exit parameter

block (RWXP). The exit does not receive a parameter list. “Entry and Return Specifications” on page 85 gives a complete list of the registers on entry to the single-record, read-write exit.

The RWXP parameter block contains the address of the CKDS or PKDS record that is being processed and information about the situation in which the exit is called. The exit sets a return code in a field in the block to specify whether the processing should continue. Table 14 describes the RWXP control block.

Table 14. RWXP Control Block Format

Offset (Dec)	Number of Bytes	Description
0	4	Name. The name of the control block. The field contains the character string `RWXP`.
4	2	Version. The version of the control block. The field contains the character string `02`.
6	2	Length. The length of the control block. The value of this field is 28 in decimal.
8	4	Return Code. The value the exit returns. Valid decimal values for this field are: **Return Code** — **Description** **0** — Process current CKDS record **4** — Do not process current CKDS record **8** — End processing
12	4	Address of the CKDS record.
16	4	The value in decimal of the length of the CKDS record.
20	7	Action. The field is a 7-byte character string describing the action performed on the CKDS record. The field can contain these values: • `READ` • `WRITE` • `DELETE` Note that the value of the field is left-justified and padded with blanks.

Table 14. RWXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
27	1	Exit Invocation Reason The reason that the exit was invoked. The field relates to only the CKDS and can contain one of the following values: **2** Refresh of the in-storage CKDS with a disk copy of a CKDS. The value of the Action field is READ. **3** Reencipher of the in-storage CKDS from a disk copy of a CKDS. The value of the Action field is READ or WRITE. **5** Conversion record postprocessing. The value of the Action field is WRITE. **8** Key entry hardware input. The value of the Action field is READ or WRITE.
28	4	Data set type (either CKDS or PKDS).

Return Codes

You can pass a return code back to the single-record, read-write process in the RWXP control block (offset +8) or in register 15. The exit can use the return code to reject records or to end the single record read-write process. The following values are valid decimal return codes:

Return Code	Description
0	Process the current CKDS record.
4	Do not process the current CKDS record.
8	End processing.

Exit Points for Security Installation Exits

IBM-supplied security exit routines were removed in ICSF/MVS Version 2 Release 1. The exit points themselves are still available.

Security Installation Exits

ICSF provides the following exit points to control access to the keys in the in-storage CKDS and to the callable services.

- Security Initialization Exit
- Security Termination Exit
- Security Service Exit
- Security Key Exit

Purpose and Use of the Exits

There are two groups of security exits. The security initialization exit (CSFESECI) and security termination exit (CSFESECT) are called during ICSF mainline processing to maintain a security communication area that is used by the other security exits. The security service exit (CSFESECS) and security key exit (CSFESECK) can be used to control access to resources on ICSF.

Below is a description of each point where ICSF calls security exit routines.

Security Initialization Exit
ICSF calls this exit during initialization just before calling the ICSF mainline exit CSFEXIT. You can use this exit to anchor resource lists, work areas, and other data to the security communication area.

Security Termination Exit
ICSF calls this exit as the last function when ICSF ends, before deleting all the installation exits. You can use this exit to free whatever is anchored to the security communication area.

Security Service Exit
ICSF calls this exit when an application uses an IBM-supplied callable service, before calling any other installation exit that is associated with that service. You can use this exit to control access to a callable service. Refer to Table 8 on page 97 for a list of callable services.

Security Key Exit
ICSF calls this exit when an application uses a key in the in-storage CKDS, before any other installation exit associated with that use of the key is called. You can use this exit to control access to the keys in the CKDS.

Environment of the Exits

The security initialization and termination exits receive control with the following characteristics:
- Supervisor state
- Key 0
- APF-authorized
- TCB mode
- Address Space Control mode=access register mode
- AMODE(31)
- RMODE(ANY)

The exits can change the characteristics during their processing. However, the exits must return to ICSF with the same characteristics as on entry.

The security service and key exits receive control with the following characteristics:
- Supervisor state
- Key 0
- APF-authorized
- TCB mode
- Cross memory mode
- AR mode
- AMODE(31)
- RMODE(ANY)

The exits can change the characteristics during their processing. However, the exits must return to ICSF with the same characteristics as on entry.

Note: The security exits are not called in SRB mode.

Installing the Exits

You install the security exits by installing the load module that contains the exit into an APF authorized library. ICSF uses the following normal search order to locate the exit:
- Job pack area
- Steplib (if one exists)

- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

Use the EXIT keyword in the installation options data set to define the ICSF name and load module name. For information about the installation options data set, see Changing Parameters in the Installation Options Data Set. The EXIT keyword has the following syntax:

EXIT (*ICSF name*, *load module name*, **FAIL (***options*))

The **ICSF name** portion of the keyword refers to the ICSF identifier for each exit, CSFESECI, CSFESECT, CSFESECS, and CSFESECK. The **load module name** is the name of the load module that contains the exit. The name can be any valid name your installation chooses. The action that the **FAIL** portion of the EXIT keyword specifies depends on the type of security exit.

For the security initialization and termination exits, the FAIL portion specifies the action ICSF takes if the exit cannot be loaded. The valid FAIL options mean:

NONE Continue initialization even if exits cannot be loaded.
SERVICE Continue initialization even if exits cannot be loaded.
EXIT Continue initialization even if exits cannot be loaded.
ICSF End ICSF if exits cannot be loaded.

You must specify a FAIL option. If you do not, ICSF returns an error message, ends abnormally, and generates an SVC dump when attempting to load the exit.

If the security initialization exit ends abnormally, ICSF ends. If the security termination exit ends abnormally, ICSF continues to end.

For the security service and key exits, the FAIL portion specifies the action ICSF takes if the exit cannot be loaded or ends abnormally. When the service or key exit is loaded, the valid FAIL options mean:

NONE Continue initialization even if exits cannot be loaded.
SERVICE Continue initialization even if exits cannot be loaded.
EXIT Continue initialization even if exits cannot be loaded.
ICSF End ICSF if exits cannot be loaded.

You must specify a FAIL option. If you do not, ICSF returns an error message, ends abnormally, and generates an SVC dump when attempting to load the exit.

When the security service exit ends abnormally, the valid FAIL options mean:

NONE Process subsequent calls to the service as if no abnormal ending occurred. Call the exit for each call of a callable service.

SERVICE Fail on subsequent calls to the particular service.

EXIT Do not call the exit again. Bypass the exit on subsequent calls to any IBM callable service.

ICSF End ICSF.

If the security service exit ends abnormally, ICSF ends the service call before performing the service.

When the security key exit ends abnormally, the valid FAIL options mean:

NONE Process subsequent attempts to access the in-storage CKDS as if no abnormal ending occurred. Call the exit for each access attempt.

SERVICE Fail on subsequent attempts to access the CKDS.

EXIT Do not call the exit again. Bypass the exit on subsequent accesses of the CKDS.

ICSF End ICSF.

If the security key exit ends abnormally, ICSF ends the attempt to access the CKDS before performing the access.

Input

The security initialization and termination exits receive the address of an 8-byte security communication area in register 1. When ICSF starts, the security initialization exit can use this area as an anchor for resource lists, work areas, or any other data that your service or keys security exits need to check authorizations. When ICSF ends, the security termination exit can free any system resources that are anchored to this area and used by the service or keys security exits. For example, the exit can free storage that is allocated from the common storage area (CSA).

When a call to a service occurs, the security service exit receives the address of an address list passed in register 1. Table 15 describes the parameters the exit receives:

Table 15. Parameters Received by the Security Service Exit

Parameter	Number of Bytes	Description
1	8	The security communication area.
2	8	The character string `CSFSERV`.
3	8	The name of the service being called.

When an attempt to access a CKDS entry occurs, the security key exit receives the address of an address list passed in register 1. Table 16 describes the parameters this exit receives:

Table 16. Parameters Received by the Security Key Exit

Parameter	Number of Bytes	Description
1	8	The security communication area.
2	8	The character string `CSFKEYS`.
3	64	The label of the key entry being accessed.

Register 0 contains the address of the exit parameter block (EXPB). See Figure 8 on page 100 and Table 10 on page 100.

Return Codes

All the security exits can pass back a return code in register 15. The security initialization exit supports the following decimal return codes:

Return Code	Description
0	Proceed with initialization.
4	End ICSF.

Any return codes other than those listed cause ICSF to end abnormally.

The security termination exit supports the following decimal return codes:

Return Code	Description
0 or 4	Proceed with termination.

Any return codes other than those listed cause ICSF to end abnormally.

The security service exit supports the following decimal return codes:

Return Code	Description
0 or 4	Proceed with the service call.

Any return codes other than those that are listed cause the service call to fail.

The security key exit supports the following decimal return codes:

Return Code	Description
0 or 4	Proceed with the access of the CKDS entry.

Any return codes other than those that are listed cause the access of the key to fail.

Key Generator Utility Program Installation Exit

The key generator utility program (KGUP) generates and maintains keys in the cryptographic key data set (CKDS). You can use KGUP to generate or supply a key to update the CKDS. KGUP generates keys to use in key exchange with other systems. ICSF provides an exit for customizing KGUP processing. For information about using KGUP to managing cryptographic keys, see *z/OS ICSF Administrator's Guide*.

Purpose and Use of the Exit

You can use the KGUP installation exit (CSFKGUP) to modify records in the CKDS, write copies of records to alternate data sets, or put additional information in the SMF record. There are many other uses for the KGUP exit depending on your installation's needs. Examine the calling points for an exit and the active control block fields at each calling point to determine other applications for the exit.

KGUP Calling Points

After an ICSF administrator submits a KGUP job for processing, KGUP calls exits at four points in processing:

1. **During KGUP initialization.** This is known as the KGUP preprocessing exit. After the KGUP job begins but before KGUP starts processing a control statement, KGUP calls this exit.

   You can use this exit to place additional information in the installation data field of the CKDS header record. You may want to do this if you need to process different cryptographic key data sets differently. You can place information in the installation data field of the record, and then subsequent calls of the exit can use this information as the basis for performing processes.
2. **Before KGUP processes a key that is identified by a control statement.** This is known as the record preprocessing exit. Before KGUP accesses the CKDS to retrieve the key that is requested in the control statement, KGUP calls the exit again.

Note: This call occurs before KGUP accesses the CKDS. If an exit routine alters a key entry at this call, KGUP accesses the CKDS with the altered entry.

You can use this exit to provide additional security for entering clear key values. When a user enters a clear key in a control statement, use the exit to change the value. In this way, the user never knows the actual clear value in the CKDS. For example, a user enters zeros for clear key values. Your exit generates some random number and replaces the user's clear key value. KGUP then processes the exit's random number as the value to write to the CKDS.

3. **Before KGUP updates the CKDS with a key entry.** This is known as the record postprocessing exit. After KGUP processes a key and before KGUP updates the CKDS, KGUP calls the exit a third time.

   At this call, the installation exit can change any information in the Key Output Data Set. Changing the Key Output Data Set also enters the changed keys into the Control Statement Output Data Set, if the keys are exportable. You can use this exit to create audit trails.
4. **During KGUP termination.** This is known as the KGUP postprocessing exit. Calls to this exit occur after KGUP completes processing but before KGUP returns control to ICSF.

Note: If an error occurs in exit processing, KGUP does not call the remaining exit invocations. If an error occurs in KGUP processing that does not result in an abnormal ending, KGUP does not call the remaining exit invocations.

Processing in the Exit

At each call, the exit receives the address of the KGUP exit parameter block (KGXP) in register 1. The exit can access any of the data in KGXP. The exit can alter some of the fields in KGXP, while others are simply references. Also, the KGUP exit can alter some fields at some calls but not at other calls.

A field in KGXP gives the calling point of the exit. The exit uses this field to determine when to call the exit to perform appropriate processing. "Input" on page 120 gives a more detailed explanation of the KGXP control block, the values it contains, and when an exit can use or change the values.

Environment of the Exit

The KGUP calls the exit only in the address space where KGUP is running. The exit receives control with the following characteristics:

- Supervisor state
- APF-authorized
- TCB mode
- Address Space Control mode=primary
- AMODE(31)
- RMODE(ANY)

The exit can change the characteristics during its processing. However, the exit must return to its caller with the same characteristics as on entry.

Installing the Exit

Install the load module that contains the exit into an APF authorized library. ICSF uses the following search order to locate the exit:

- Job pack area
- Steplib (if one exists)

- Joblib (if one exists)
- Link pack area (LPA)
- Link list (SYS1.LINKLIB concatenation)

Define the ICSF name and load module name on the EXIT keyword in the installation options data set. For more information about the installation options data set, see “Changing Parameters in the Installation Options Data Set” on page 22. The EXIT keyword has the following syntax:

EXIT (*ICSF name***,** *load module name***, FAIL (***options***))**

The **ICSF name** portion of the keyword refers to the ICSF name for the KGUP exit. The ICSF name for the KGUP exit is CSFKGUP. The **load module name** is the name of the load module that contains the exit. The name can be any valid name that your installation chooses. The **FAIL** portion of the EXIT keyword specifies the action ICSF takes if the exit cannot be loaded. The valid FAIL options are **NONE, EXIT, SERVICE,** and **CSF.** The FAIL options available to the KGUP exit are the following:

NONE Initialization continues even if exit cannot be loaded.
ICSF Initialization ends if exit cannot be loaded.

You must specify a FAIL option. If you do not, ICSF returns an error message, ends abnormally, and generates an SVC dump when attempting to load the exit. If the exit ends abnormally, KGUP also ends abnormally.

Input

At each of the invocation points, the exit receives the address of the KGUP exit parameter block (KGXP) in register 1. The exit does not receive a parameter list. “Entry and Return Specifications” on page 85 gives a complete list of the registers on entry to the KGUP exit.

The KGUP exit can alter some of the fields in KGXP. Some fields only provide information to the exit and cannot be changed, and some fields do not apply to particular calls to the exit.

Table 17 describes the KGXP control block.

Table 17. KGXP Control Block Format

Offset (Dec)	Number of Bytes	Description
0	4	Block Identifier. The name of the control block. The field must contain the character string `KGXP`. The exit must not change the value and KGUP does not use the field upon return from the exit.
4	2	Block Version Number. The version of the control block. The field must contain the character string `02`. The exit cannot change this field and KGUP does not use this field on return from the exit.
6	2	Block Length. The length of the control block. The decimal value of the field is 172. The exit cannot change the field and KGUP does not use this field on return from the exit.

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
8	4	Return Code. The return code the exit supplies upon completion. Upon entry, KGUP initializes this field to zeros. The valid decimal return codes for each of the invocation points are: Record Pre- or postprocessing. **0** Normal, continue processing. **4** Reject control statement, but do not end KGUP. **8** End KGUP immediately. KGUP pre- or postprocessing. **0** Normal, continue processing. **> 0** End KGUP immediately.
12	1	Call Point. Indicates the invocation point of the exit. The exit cannot change this field and KGUP does not use this field on return from the exit. You can determine the invocation point by the bit that is set on. **Bit** **Meaning When Set On** **0** KGUP preprocessing invocation. **1** KGUP postprocessing invocation. **2** Record preprocessing invocation. **3** Record postprocessing invocation. **4-7** **Reserved**.
13	1	Options. Indicates the keywords specified on the KGUP control statement. The exit cannot change this field and KGUP does not use the field upon return from the exit. The field is used only during the record preprocessing and postprocessing invocations. You can determine the keywords on the control statement by the bits that are set on. **Bit** **Meaning When Set On** **0** LABEL with multiple values specified. **1** RANGE specified. **2** KEY specified. **3** CLEAR specified. **4** SINGLE specified. **5** NOCV specified. **6** OUTTYPE specified. **7** **Reserved.**

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
14	1	Verb Type. Indicates the verb used on the KGUP control statement. The exit cannot change this field and KGUP does not use this field on return from the exit. The field is used only for the record preprocessing and record postprocessing invocations. You can determine the verb on the control statement by the bit that is set on. **Bit** — **Meaning When Set On** **0** — ADD **1** — UPDATE **2** — DELETE **3** — RENAME **4** — SET **5–7** — **Reserved.**
15	1	KGUP Flags. Indicates the processing conditions encountered by KGUP at the record postprocessing invocation. The exit cannot change this field and KGUP does not use the field upon return from the exit. The field is not used for the KGUP pre- or postprocessing invocations or the record preprocessing invocation. The processing conditions can be determined by examining whether bit 0 is set on. **Bit** — **Meaning When Set On** **0** — Non-odd parity key was imported. **1–7** — **Reserved.**

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
16	72	Action Key. Contains the key index accessed by the KGUP control statement. The key index consists of the key label and type fields of a CKDS record entry (“Debugging Aids” on page 144 describes the CKDS record format in greater detail). The key index is the first 72 bytes of a CKDS record, and the information in the key index is used to differentiate one key from another. The exit can modify the field at the record preprocessing invocation. The field is not used for the KGUP pre- or postprocessing invocation or the record postprocessing invocation. If the exit modifies the field, KGUP uses the modified field to access the CKDS upon return from the exit. Before the record preprocessing invocation, KGUP places the following in this field: • The key label or key old label from the LABEL or key label from the RANGE keyword of the control statement • The key type from the TYPE keyword of the control statement The exit cannot modify the key label, key old label, or key type.
88	72	Rename Key. Contains the key index used to rename a key when RENAME is the verb on the control statement. The key index consists of the key label and type fields of a CKDS record entry. The exit can modify the field at the record preprocessing invocation. The field is not used for the KGUP pre- or postprocessing or record postprocessing invocations. If the exit modifies the field, KGUP uses the modified field to access the CKDS upon return from the exit. Before the record preprocessing invocation, KGUP places the following information in this field: • The key new label from the LABEL keyword of the control statement. • The key type from the TYPE keyword of the control statement. The exit cannot modify the key new label or the key type.

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
160	72	Transkey key-label1. The key index of the TRANSKEY key-label1 on the KGUP control statement. The key index is the key label and type of the CKDS record entry. The exit can modify the field at the record preprocessing invocation. The field is not used for the KGUP pre- or postprocessing and record postprocessing invocations. If the exit modifies the field, KGUP uses the modified field to access the CKDS upon return from the exit. Before the record preprocessing invocation, KGUP places the following information in this field: • The key-label1 from the TRANSKEY keyword of the control statement. • The key type. The type is IMPORTER, if keys are supplied; the type is EXPORTER, if keys are not supplied. The exit cannot modify the key-label1 or the key type.
232	72	Transkey key-label2. The key index of the TRANSKEY key-label2 on the KGUP control statement. The key index is the key label and type of the CKDS record entry. The exit can modify the field at the record preprocessing invocation. The field is not used for the KGUP pre- or postprocessing and record postprocessing invocations. If the exit modifies the field, KGUP uses the modified field to access the CKDS upon return from the exit. Before the record preprocessing invocation, KGUP places the following information in this field: • The key-label2 from the TRANSKEY keyword of the control statement. • The key type. The key type is IMPORTER, if keys are supplied; the type is EXPORTER, if keys are not supplied. The exit cannot modify the key-label2 or the key type.
304	8	The OUTTYPE value, if specified. If no OUTTYPE is specified, this field set to binary zeros.

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
312	16	Key key-value. The value of the key supplied on the KGUP control statement. The 16 bytes are hexadecimal characters representing the 8-byte hexadecimal key value. The field contains a value only if the KEY option was specified and a key value was supplied on the control statement. You can determine whether the KEY option was used by examining bit 2 at offset +13 in KGXP. If TRANSKEY was specified on the control statement, KGUP decrypts key-value1 under the transport key specified with the TRANSKEY keyword. If CLEAR was specified on the control statement, KGUP does not decrypt key-value1. The exit can modify the field at the record preprocessing invocation. This field is not used for the KGUP pre- or postprocessing invocations or the record postprocessing invocation. The field does not contain a value when generating keys. The exit is permitted to put values in this field only if a key was supplied on the control statement. The exit-supplied value must be edited for hexadecimal values and it then replaces the values entered on the input control statement.
328	16	Key ikey-value. The value of the intermediate key supplied on the KGUP control statement. The 16 bytes are hexadecimal characters representing the 8-byte hexadecimal key value. The field contains a value only if the KEY option was specified and a key value was supplied on the control statement. You can determine whether the KEY option was used by examining bit 2 at offset +13 in KGXP. If TRANSKEY was specified on the control statement, KGUP decrypts the ikey-value under the transport key specified with the TRANSKEY keyword. If SINGLE was specified on the control statement, the ikey-value will be equal to the key-value. If CLEAR was specified on the control statement, KGUP does not decrypt the ikey-value. The exit can modify the field at the record preprocessing invocation. This field is not used at the KGUP pre- or postprocessing invocation or the record postprocessing invocation. The field does not contain a value when generating keys. The exit can put values in this field only if a key was supplied on the control statement. The exit-supplied value must be edited for hexadecimal values; it then replaces the values entered on the input control statement.

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
344	16	Key 3key-value. The value of the third key supplied on the KGUP control statement. The 16 bytes are hexadecimal characters representing the 8-byte hexadecimal key value. The field contains a value only if the KEY option was specified and a key value was supplied on the control statement. You can determine whether the KEY option was used by examining bit 2 at offset +13 in KGXP. If TRANSKEY was specified on the control statement, KGUP decrypts the 3key-value under the transport key specified with the TRANSKEY keyword. If CLEAR was specified on the control statement, KGUP does not decrypt the 3key-value. The exit can modify the field at the record preprocessing invocation. This field is not used at the KGUP pre- or postprocessing invocation or the record postprocessing invocation. The field does not contain a value when generating keys. The exit can put values in this field only if a key was supplied on the control statement. The exit-supplied value must be edited for hexadecimal values; it then replaces the values entered on the input control statement.
360	4	CSFKEYS record for transkey, key-label1. The address of the CSFKEYS data set record that is output for transkey key-label1 on the KGUP control statement. The field ONLY contains a value when generating keys. This field is filled in when CLEAR keys are generated. The exit can modify the field at the record postprocessing invocation. KGUP sets the address to zero for the KGUP pre- or postprocessing and record preprocessing invocations. KGUP does not check the field upon return from the exit. Normal CSFKEYS processing applies. KGUP uses key values on control statement creation. For the format of the CSFKEYS record, refer to *z/OS ICSF Administrator's Guide*.
364	4	CSFKEYS record for transkey, key-label2. The address of the CSFKEYS data set record that is output for transkey key-label2 on the KGUP control statement. The field ONLY contains a value when generating keys. This field is only filled in when TRANSKEY key-label2 is specified for generated keys. The exit can modify the field at the record postprocessing invocation. KGUP sets the address to zero for the KGUP pre- or postprocessing and record preprocessing invocations. KGUP does not check the field upon return from the exit. Normal CSFKEYS processing applies. KGUP uses key values on control statement creation.

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
368	4	CSFCKDS header record. The address of the CSFCKDS data set header record. The exit can check the field at the KGUP pre- or postprocessing invocations. However, the exit can modify the field only at the KGUP postprocessing invocation. KGUP sets the value of the field to zero for the record pre- or postprocessing invocations. The exit can modify the installation data field of the CKDS header record (see “Debugging Aids” on page 144 for a description of the CKDS header record. Offset +196 of the CKDS header record is the installation data field). The installation data field supplied by the exit is placed in the CKDS header record after the KGUP postprocessing invocation returns control to KGUP.
372	4	CSFCKDS record. The address of the CSFCKDS data set record processed by the KGUP control statement. KGUP sets the address to zero if the TRANSKEY keyword has two labels of transport keys. The exit can check the field only at the record postprocessing invocation. KGUP sets the address to zero for the record preprocessing and KGUP pre- or postprocessing invocations. The exit can modify the record area if the TRANSKEY keyword does not have two labels.
376	4	RENAME CSFCKDS record. The address of the CSFCKDS data set record processed when the RENAME verb is used in a control statement. You can determine whether the RENAME verb was used by examining bit 3 at offset +14 in KGXP. The exit can modify the field at the record postprocessing invocation. KGUP sets the address to zero for the record preprocessing and KGUP pre- or postprocessing invocations. The exit can modify the record area. KGUP does not check this field upon return from the invocation. Normal CSFCKDS processing applies.
380	4	Installation data. The address of the data specified on the INSTDATA keyword of the KGUP control statement. The address of the area is zero if a SET control statement has not been processed. “The SET Statement” on page 128 describes how to use the field in greater detail.

Table 17. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description
384	4	Installation exit area. The address of an area set by the installation that is preserved across all invocations of the exit. The first byte of the area contains the length of the area (including the length byte). After KGUP completes, the first 64 bytes of the area are written to the SMF data set. The exit has exclusive control of modifying this area. The area is only used as input to SMF processing upon completion of KGUP.

The SET Statement

Use the SET control statements to specify data to send to a KGUP installation exit. For a more detailed description of the SET statement, see *z/OS ICSF Administrator's Guide*.

The installation data field in KGXP (offset +364) contains the address of the data SET statement specifies. Data that is specified on a SET statement can be especially useful if you alter key entries. You may want to keep track of the entries you change by putting the original data and the changed data in the installation data area.

Return Codes

You can pass a return code back to KGUP in the KGXP control block (offset +8). The exit can use the return code to cause KGUP to reject control statements or to end KGUP. Return code values, in decimal, for record pre- or postprocessing exit calls are:

Return Code	Description
0	Normal, continue processing.
4	Reject control statement, but do not end KGUP.
8	End KGUP.

All other return codes are not valid and cause KGUP to end.

Return code values, in decimal, for the KGUP pre- or postprocessing invocations are:

Return Code	Description
0	Normal, continue processing.
>0	End KGUP.

Chapter 8. Operating ICSF

You use certain commands to operate ICSF. Also, there are different conditions for operating ICSF that you should consider. This chapter describes the ICSF operating tasks.

Starting and Stopping ICSF

If you running with FMID HCR7708 on an IBM @server zSeries 990, see Appendix E, “Running HCR7708 on a IBM @server zSeries 990”, on page 195.

To start ICSF, issue the operator START command. You must issue the START command after each IPL. When you issue the START command, verification tests check that the master key in each coprocessor is the same as the master key that enciphered the cryptographic key data set (CKDS). Verification tests are also performed to ensure that the PCI Cryptographic Coprocessor SYM-MK on all PCI Cryptographic Coprocessors is the same as the DES master key on the Cryptographic Coprocessor Features, and that the PCI Cryptographic Coprocessor ASYM-MK on all PCI Cryptographic Coprocessors is the same as the PKA SMK on the Cryptographic Coprocessor Features. If a master key does not match the CKDS, the following occurs:

- ICSF starts.
- A message that indicates the verification failed for the indicated coprocessor and CPU appears on the console.
- The Cryptographic Coprocessor Feature on the CPU for which the verification failed is not active.

When ICSF successfully starts, a message that indicates that initialization is complete appears on the console.

The following example shows the format of the START command to start ICSF, assuming that CSF is the name of the start procedure:

```
START CSF
```

You can start ICSF only as a started task.

To stop ICSF, issue the operator STOP command. After you issue the command, all ICSF processing stops. If ICSF stops successfully, a message that states that ICSF is stopped appears on the console.

The following example shows the format of the STOP command to stop ICSF, assuming that CSF is the name of the started procedure:

```
STOP CSF
```

Note: In general, you should *not* use the operator CANCEL command to end CSF operation. Issuing the CANCEL command does not take the cryptographic feature offline. You can cancel CSF and then restart CSF. This is unlike the Programmed Cryptographic Facility (PCF), which cannot be stopped and restarted.

Modifying ICSF

When you issue the MODIFY command, ICSF gives control to the installation exit CSFEXIT5, if it exists. Your installation can write an exit routine for CSFEXIT5 that changes ICSF operations. For example, you might have the installation exit change the CHECKAUTH and KEYAUTH installation options without having to stop and restart ICSF. See Chapter 7, “Installation Exits”, on page 83 for a description of the installation exits.

If your installation does not write an exit routine for CSFEXIT5, no action occurs when you enter the MODIFY command.

Using Different Configurations

A central processor complex can have up to two cryptographic features. If you have a processor complex with more than one cryptographic feature, you can configure the complex to run in one of several modes, depending on your central processor hardware. This section describes the different configurations available.

Configuring the S/390 Enterprise Servers, the S/390 Multiprise Server, the IBM @server zSeries 800 and the IBM @server zSeries 900

The Cryptographic Coprocessor Feature can include up to two cryptographic coprocessors, each of which is attached to a central processor within the central processor complex. Each cryptographic coprocessor, or crypto CP, has sixteen master key register sets, referred to as domains. ICSF uses the domain usage index to access each domain. You can configure the complex to run in one of the following modes:

- Single image mode
- Logical partition mode

Single Image Mode

In single image mode, the processor complex has access to the same domain on both Crypto CP 0 and Crypto CP 4. The domain is specified in the installation options data set. Beginning in z/OS V1 R2, the DOMAIN parameter is optional. It is required if more than one domain is specified as the usage domain on the PR/SM panels or if running in native mode. See “Changing Parameters in the Installation Options Data Set” on page 22 for additional information on the DOMAIN parameter. The accessed domain on both coprocessors must have the same master key. Figure 9 shows an example single image mode configuration.

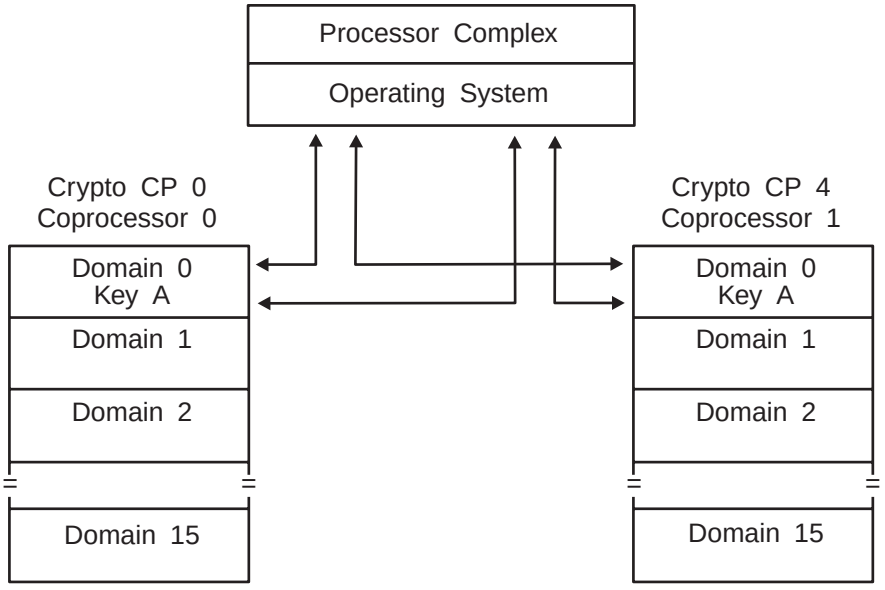


Figure 9. Two Crypto CPs on a Processor Complex Running in Single Image Mode

Logical Partition (LPAR) Mode

You can divide your processor complex into PR/SM logical partitions (LPARs). When you create logical partitions on your processor complex, you use the usage domain index on the Support Element Customize Image Profile page to enable access to a Crypto CP domain. The number that is specified for the usage domain index must correspond to the domain number you specify with the DOMAIN(n)

keyword in the installation options data set. Beginning in z/OS V1 R2, the DOMAIN parameter is optional. It is required if more than one domain is specified as the usage domain on the PR/SM panels or if running in native mode. See “Changing Parameters in the Installation Options Data Set” on page 22 for additional information on the DOMAIN parameter.

You can assign more than one domain to an LPAR, but you must use a unique installation options data set to define each domain. Assigning more than one domain to an LPAR makes it possible to have separate master keys for different purposes. For example, you might have one master key for production operations and a different master key for test operations.

The PCI Cryptographic Coprocessor can be configured like a Cryptographic Coprocessor Feature. It can be dedicated or shared across multiple partitions with each card supporting up to 16 domains.

The PCI Cryptographic Accelerator can be configured like a Cryptographic Coprocessor Feature. It can be dedicated or shared across multiple partitions with each card supporting up to 16 domains.

When using logical partitions, there is no domain sharing unless TKE is being used. The ’HOST’ LPAR can control the domains of the other LPARS if the control domain for the first LPAR is setup for it. The example in Figure 10 on page 133 shows that LPAR 1 has access to Domain 0 on Crypto CP 0, Crypto CP 1, and PCICC. LPAR 2 has access to Domain 1 and Domain 2 on both Crypto CPs and on the PCICC. LPAR 1 cannot access Domain 1 or Domain 2 on the PCICC or on either of the Crypto CPs. Likewise, LPAR 2 cannot access Domain 0 on either Crypto CP or the PCICC. The ICSF system running on the operating system in LPAR 2 has access to only one domain at any particular time, as specified in the installation options data set.

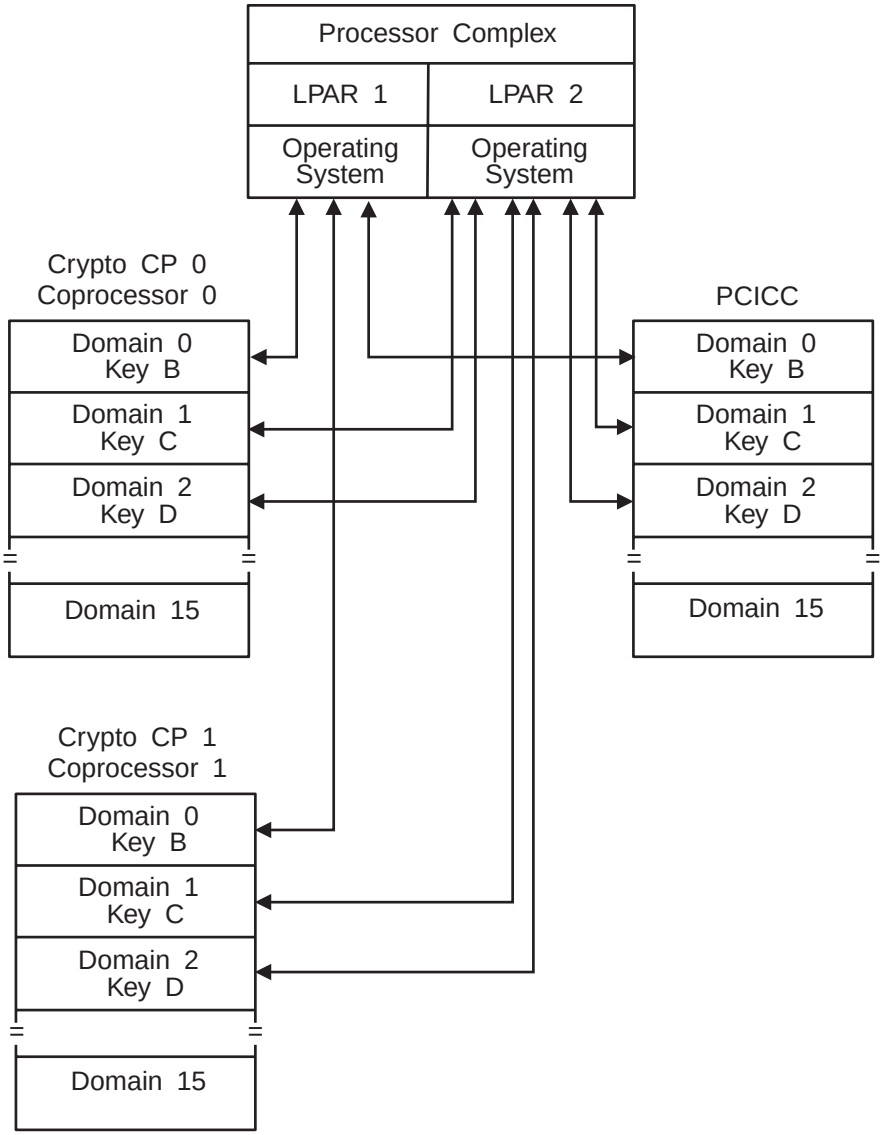


Figure 10. Three Crypto CPs on a Processor Complex Running in LPAR Mode

Configuring the IBM @server zSeries 990

There is only LPAR mode on an IBM @server zSeries 990. You can divide your processor complex into PR/SM logical partitions. When you create logical partitions on your processor complex, you use the usage domain index on the Support Element Customize Image Profile page only if you have PCICAs or plan to add PCICAs to your system.

The DOMAIN parameter is optional. The number that is specified for the usage domain index must correspond to the domain number you specified with the DOMAIN(n) keyword in the installation options data set – if you specified one. The DOMAIN keyword is required if more than one domain is specified as the usage domain on the PR/SM panels.

The PCI Cryptographic Accelerator can be configured and shared across multiple partitions.

With the IBM @server zSeries 990, there is support for 30 LPARs. On previous systems, where the maximum number of LPARs was 16, a domain was unique to an LPAR. With more than 16 LPARs to support, the domain may not be unique across LPARs but the same domain may be assigned to different LPARs if they are accessing different PCICAs. This is illustrated by LPAR 1 and LPAR 3 in Figure 11. They are both assigned to usage domain 0 but on two different PCICAs.

The example in Figure 11 shows that LPAR 2 has assigned access to Domain 1 on both PCICA 1 and PCICA 2. If you were to add another PCICA, Domain 1 on the new PCICA would also be assigned.

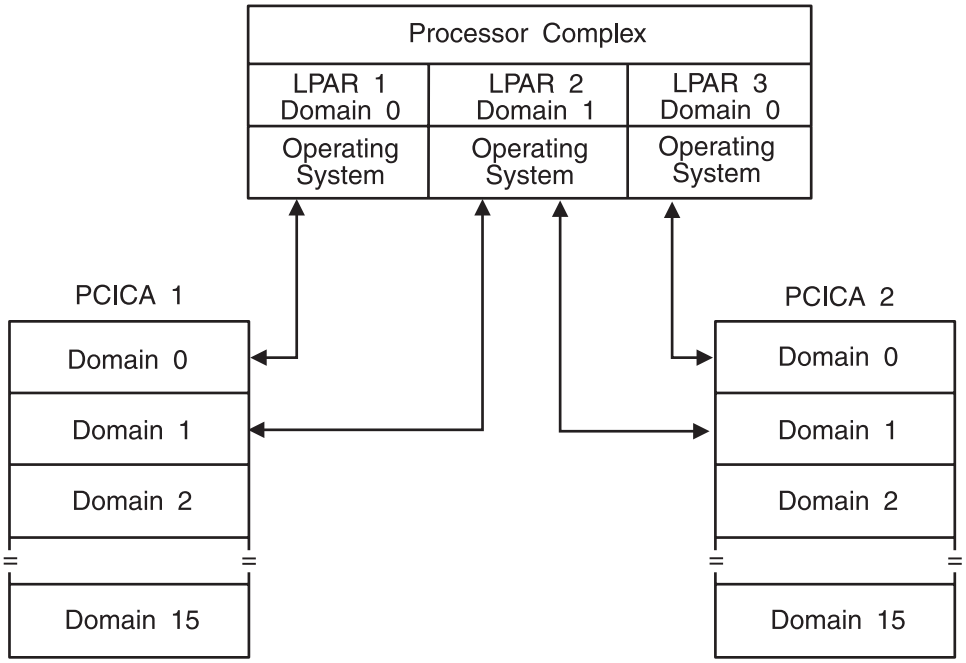


Figure 11. Two Crypto PCICAs on a Processor Complex Running in LPAR Mode

Disabling Cryptographic Coprocessors

For S/390 Enterprise and S/390 Multiprise servers that support TKE, cryptographic functions can be disabled through the ECM function. This places the Cryptographic Coprocessor Feature in standby mode. The functions can be brought out of standby mode by enabling the cryptographic function bit in the ECM through TKE.

Note: With z/OS V1 R3, status displayed on the hardware status panel will no longer be STANDBY. It will now show DISABLED.

The Coprocessor Management Panel allows you to deactivate/activate a PCI Cryptographic Coprocessor or PCI Cryptographic Accelerator. With TKE V3.0 or higher, you can also disable/enable the PCI Cryptographic Coprocessor or PCI Cryptographic Accelerator. When a PCI Cryptographic Coprocessor or PCI Cryptographic Accelerator is deactivated through the Coprocessor Management Panel, the card is only deactivated for that one LPAR. When a PCI Cryptographic Coprocessor or PCI Cryptographic Accelerator is disabled by TKE, the card is disabled for the entire system, not just the LPAR that issued the disable.

Performance Considerations for Using Installation Options

You specify installation options in the installation options data set. Two installation options, CHECKAUTH and KEYAUTH, provide additional security checking, but affect performance.

In ICSF, the Security Server (RACF) always checks non-Supervisor State callers. The CHECKAUTH option allows you to specify whether CSF performs access control checking of Supervisor State and System Key callers. Specify CHECKAUTH(NO) if you do not want CSF to check Supervisor State and System Key callers. Specify CHECKAUTH(YES) if you want CSF to check Supervisor State callers. Checking Supervisor State and System Key callers significantly affects performance.

The KEYAUTH option allows you to specify whether ICSF should authenticate an entry in the CKDS whenever ICSF accesses the entry. ICSF creates a message authentication code (MAC) for each entry in the CKDS and stores the MAC with the entry. Whenever ICSF retrieves an entry from the CKDS, ICSF uses the MAC to authenticate the entry. When ICSF authenticates the entry, ICSF verifies that the entry was not inadvertently changed or damaged. If the authentication fails, ICSF returns either a return code with a reason code or message.

You specify KEYAUTH(NO) for ICSF not to authenticate an entry or KEYAUTH(YES) for ICSF to authenticate an entry. The authentication has a small impact on performance. The chance of an error occurring in the in-storage CKDS is minimal. However, the authentication might be useful for diagnostic purposes if an error occurs.

See “Create the Installation Options Data Set” on page 14 and “Changing Parameters in the Installation Options Data Set” on page 22 for more information.

VTAM Session-Level Encryption

ICSF supports VTAM session-level encryption. VTAM session-level encryption provides protection for messages within SNA sessions; that is, between pairs of logical units that support their respective end users. When this method of protection is in effect, data is enciphered by the originating logical unit and deciphered only by the destination logical unit. Thus, the data never appears in the clear while passing through the network.

Restriction: Running with FMID HCR7708 on an IBM @server zSeries 990 is not supported.

ICSF places no restrictions on the addressing mode of calling programs. In particular, when VTAM session-level encryption is used with ICSF, VTAM can use storage above 16 megabytes.

System SSL Encryption

ICSF supports System SSL encryption on the IBM @server zSeries 900, IBM @server zSeries 800,and S/390 G6 Enterprise Server. On the IBM @server zSeries 990, a PCI Cryptographic Accelerator is required.

On CCF systems that also have PCICAs, you can run system SSL encryption without entering master keys.

Access Method Services Cryptographic Option

In compatibility mode, ICSF supports the Access Method Services Cryptographic Option. The option enables the user of the Access Method Services REPRO command to use the Data Encryption Algorithm to encipher data.

Restriction: Running with FMID HCR7708 on an IBM @server zSeries 990 is not supported.

The Access Method Services user can use REPRO to encipher data that is written to a data set, and then store the enciphered data set offline. When desired, you can bring the enciphered data set back online, and use REPRO to decipher the enciphered data. You can decipher the data either on the host processor on which it was enciphered, or on another host processor that contains the Access Method Services Cryptographic Option and the same cryptographic key that was used to encipher the data. You can either use ICSF to create the cryptographic keys, or use keys that the Access Method Services user supplies.

With the exception of catalogs, all data set organizations that are supported for input by REPRO are eligible as input for enciphering. Similarly, with the exception of catalogs, all data set organizations supported for output by REPRO are eligible as output for deciphering. The resulting enciphered data sets are always sequentially organized (SAM or VSAM entry-sequenced data sets).

See Appendix D, “Using AMS REPRO Encryption”, on page 193 for more information in using this method.

Event Recording

ICSF records certain ICSF events in the System Management Facilities (SMF) data set. ICSF also sends messages that are generated during processing to diagnostic data sets and consoles. The SMF recording and messages help you detect problems and track events. This chapter describes the events that ICSF records in the SMF record and describes where ICSF sends certain messages.

System Management Facilities (SMF) Recording

ICSF uses SMF record type 82 to record certain ICSF events. Record type 82 contains a fixed header section and subtypes. Each subtype contains information about the event that caused ICSF to write to the SMF record.

You can map record type 82 by using the CSFSMF82 macro. *z/OS MVS System Management Facilities (SMF)* describes the format of record type 82.

ICSF records information in the SMF data set when the following events occur:

- ICSF starts
- ICSF status changes on a processor
- ICSF handles error conditions for Cryptographic Coprocessor Feature failure or tampering
- You enable or disable special secure mode
- You enter a master key part
- You use the ICSF panels to process an operational key loaded using the TKE workstation

- TKE commands and responses are all audited through SMF 82 (TKE commands on the Cryptographic Coprocessor Feature are CSFPKSC. TKE commands on the PCI Cryptographic Coprocessor use CSFPCI.)
- The in-storage cryptographic key data set (CKDS) is refreshed
- A dynamic change is made to the PKDS
- You use the ICSF panels to update the new master key register on a PCI Cryptographic Coprocessor
- You create or delete a retained key on a PCI Cryptographic Coprocessor
- The TKE workstation issues a PCI Cryptographic Coprocessor command request or receives a reply response from a PCI Cryptographic Coprocessor
- ICSF records processing times for PCI Cryptographic Coprocessors
- A PCI Cryptographic Coprocessor is either brought on line or taken off line

Each of these events causes ICSF to record information in a separate subtype in the SMF record.

Recording and Formatting type 82 SMF Records in a Report - Beginning in z/OS V1 R3, the following sample jobs are available (in SYS1.SAMPLIB) to assist in the recording and formatting of type 82 SMF data:

- **CSFSMFJ** - JCL that executes the code to dump and format SMF type 82 records for ICSF. Before executing the JCL, you need to make modifications to the JCL (see the prologue in the sample for specific instructions). After the JCL has been modified, terminate SMF recording of the currently active dump dataset (by issuing I SMF) to allow for the unloading of SMF records. After SMF recording has been terminated, execute the JCL. The output goes into the held queue. The following is an example of CSFSMFJ.

```
//CSFSMFJ  JOB <CARD PARAMETERS>
//*******************************************************************
//*  LICENSED MATERIALS - PROPERTY OF IBM                           *
//*  5694-A01                                                       *
//*  (C) COPYRIGHT IBM CORP. 2002                                   *
//*                                                                 *
//*  This JCL reads Type 82 SMF records and formats them in a report.*
//*                                                                 *
//*  CAUTION: This is neither a JCL procedure nor a complete JOB.   *
//*  Before using this JOB step, you will have to make the following *
//*  modifications:                                                 *
//*                                                                 *
//*   1) Add the job parameters to meet your system requirements.   *
//*   2) Change the DUMPIN DSN=hlq.smfdata.input to be the name of  *
//*      the dataset where you currently have SMF data being        *
//*      recorded.                                                  *
//*   3) Change the STEPLIB VOL=SER=ttttt1 and VOL=SER=ttttt2 to    *
//*      be the volumes where these sort datasets reside.           *
//*   4) Change the SYSPROC DSN=hlq.rexx.dataset to be the name of  *
//*      the dataset where you have placed the CSFSMFR REXX sample. *
//*                                                                 *
//*   Prior to executing this job, you need to terminate SMF        *
//*   recording of the currently active dump dataset for allow the  *
//*   unload of SMF records.                                        *
//*                                                                 *
//*******************************************************************
//*
//*-----------------------------------------------------------------*
//*   UNLOAD SMF 82 RECORDS FROM VSAM TO VBS                        *
//*-----------------------------------------------------------------*
//SMFDMP   EXEC  PGM=IFASMFDP
//DUMPIN   DD    DISP=SHR,DSN=hlq.smfdata.input
//DUMPOUT  DD    DISP=(NEW,PASS),DSN=&&VBS,UNIT=3390,
```

```
//         SPACE=(CYL,(1,1)),DCB=(LRECL=32760,RECFM=VBS,BLKSIZE=4096)
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
    INDD(DUMPIN,OPTIONS(DUMP))
    OUTDD(DUMPOUT,TYPE(82))
//*
//*--------------------------------------------------------------------*
//*   COPY VBS TO SHORTER VB AND SORT ON DATE/TIME                     *
//*--------------------------------------------------------------------*
//COPYSORT EXEC PGM=SORT,REGION=6000K
//STEPLIB  DD DISP=SHR,DSN=SYS1.SORTLPA,VOL=SER=ttttt1,UNIT=3390
//         DD DISP=SHR,DSN=SYS1.SICELINK,VOL=SER=ttttt2,UNIT=3390
//SYSOUT   DD SYSOUT=*
//SORTWK01 DD UNIT=3390,SPACE=(CYL,10)
//SORTIN   DD DISP=(OLD,DELETE),DSN=&&VBS
//SORTOUT  DD    DISP=(NEW,PASS),DSN=&&VB,UNIT=3390,
//         SPACE=(CYL,(1,1)),DCB=(LRECL=3000,RECFM=VB)
//SYSIN    DD *
 SORT FIELDS=(11,4,D,7,4,D),FORMAT=BI,SIZE=E4000
//*
//*--------------------------------------------------------------------*
//*   FORMAT TYPE 82 RECORDS                                           *
//*--------------------------------------------------------------------*
//FMT     EXEC PGM=IKJEFT01,REGION=5128K,DYNAMNBR=100
//SYSPROC  DD DISP=SHR,DSN=hlq.rexx.dataset
//SYSTSPRT DD SYSOUT=*
//INDD     DD DISP=(OLD,DELETE),DSN=&&VB
//OUTDD    DD SYSOUT=*
//SYSTSIN DD *
  %CSFSMFR
```

- **CSFSMFR** - An EXEC that formats the SMF records into a readable report.

ICSF Initialization (Subtype 1)

When ICSF starts, ICSF writes to subtype 1 after initialization is completed. Subtype 1 describes the values of installation options that are specified in the installation options data set.

Subtype 1 contains the following information:

- Special secure mode (SSM) option
- Key authentication (KEYAUTH) option
- Security Server (RACF) checking of Supervisor State and System Key callers (CHECKAUTH) option
- Compatibility mode with CUSP or PCF (COMPAT) option
- Cryptographic domain number (DOMAIN) option
- Number of trace entries (TRACEENTRY) option
- CKDS name (CKDSN) option
- Maximum length for data in a callable service (MAXLEN) option

  Beginning with z/OS V1 R2, the MAXLEN parameter may still be specified in the options data set, but only the maximum value limit will be enforced (2147483647). If a value greater than this is specified, an error will result and ICSF will not start.
- Compatibility encryption algorithm (COMPENC) option
- User parameter (USERPARM) option
- PKDS name (PKDSN) option

ICSF Status Change (Subtype 3)

ICSF writes to subtype 3 when processors are verified at initialization, after a master key is set or changed, when ICSF switches from stand-by mode to normal

mode, or when a processor comes online or offline. When processor status changes, subtype 3 gives the status of the processors still online.

Subtype 3 contains the following information:
- Processor number
- Coprocessor number
- Cryptographic domain number
- Master key version number

If a master key change or set occurs, subtype 3 also contains the following information:
- Master key verification pattern
- Old master key verification pattern, if an old master key exists
- New master key verification pattern, if a new master key exists

Error Handling for Cryptographic Coprocessor Feature (Subtype 4)
ICSF writes to subtype 4 when the Coprocessor is in standby mode or when the Cryptographic Coprocessor Feature detects tampering.

Subtype 4 contains the following information:
- Status word from the Cryptographic Coprocessor Feature
- Processor number
- Cryptographic domain number

Special Secure Mode Change (Subtype 5)
Subtype 5 contains special secure mode status bit. ICSF writes to subtype 5 when the status of special secure mode changes. ICSF also updates subtype 5 when the Cryptographic Coprocessor Feature indicates that special secure mode was required for an instruction, but was not enabled.

Master Key Part Entry (Subtype 6)
ICSF writes to subtype 6 when master key parts are entered using TKE workstation and are processed using the TKE master key entry ICSF panels. Subtype 6 contains the following information:
- The verification pattern for the master key part
- A bit indicating whether the verification pattern is valid
- The Coprocessor number
- The cryptographic domain number

If you enter the final master key part, the record also contains the verification pattern for the entire master key and a bit indicating whether the verification pattern is valid.

Operation Key Part Entry (Subtype 7)
ICSF writes to subtype 7 when key parts are entered using the TKE workstation and are processed using the operational key entry ICSF panels. Subtype 7 contains the following information:
- The verification pattern for the key part
- A bit indicating whether the verification pattern is valid
- The cryptographic domain number
- The Coprocessor number
- The name of the CKDS that contains the entry with the key part
- The label of the CKDS entry that contains the key part

CKDS Refresh (Subtype 8)

ICSF writes to subtype 8 when the in-storage CKDS is successfully refreshed. ICSF refreshes the in-storage CKDS by reading a disk copy of a CKDS into storage. Subtype 8 contains the following information:

- Name of the current in-storage CKDS that ICSF refreshes
- Name of the disk copy of the CKDS that ICSF read into storage to replace the current CKDS

Dynamic CKDS Update (Subtype 9)

ICSF writes to subtype 9 when an application uses the dynamic CKDS update services to write to the CKDS. Subtype 9 contains the following information:

- Name of the changed CKDS
- The operation performed
- The CKDS entry (which includes the label name and key type) that was changed

PKA Key Part Entry (Subtype 10)

ICSF writes to subtype 10 when you use the ICSF panels to enter PKA master key parts. Subtype 10 contains the following information:

- An indication of which PKA Master key is changing; the Signature Master Key (SMK), or the Key Management Master Key (KMMK)
- An indication of whether the following hash pattern of the PKA master key register is valid (It is valid when the final key part is entered.)
- The hash pattern (MDC-4) of the PKA master key register
- The hash pattern of PKA key part
- The Coprocessor number
- Current cryptographic domain

If no DES master key has been validated, the key part entries do not contain a hash pattern. The record for the final key contains the hash pattern of the complete key.

Clear New Master Key Part Entry (Subtype 11)

ICSF writes to subtype 11 when you use the ICSF panels to enter new master key parts. Subtype 11 contains the following information:

- An indication of whether the following hash pattern of the new master key master key register is valid (It is valid when the final key part is entered.)
- An indication of whether the following verification pattern of the new master key master key register is valid (It is valid when the final key part is entered.)
- The hash pattern of the new master key register
- The verification pattern of the new master key register
- The hash pattern of new master key part
- The verification pattern of new master key part
- The Coprocessor number
- Current cryptographic domain

If no DES master key has been validated, the key part entries do not contain a verification pattern and hash pattern. The record for the final key contains the verification pattern and hash pattern of the complete key.

PKSC Commands (Subtype 12)

ICSF writes to subtype 12 for every PKSC command entered through the CSFPKSC interface. Subtype 12 contains the following information:

- The complete PKSC request
- The corresponding PKSC response

Dynamic PKDS Update (Subtype 13)

ICSF writes to subtype 13 when an application uses the dynamic PKDS update services to change the PKDS. Subtype 13 contains the following information:

- The name of the changed PKDS
- The operation performed
- The name of the changed entry in the PKDS

PCI Cryptographic Coprocessor Clear Master Key Entry (Subtype 14)

ICSF writes to subtype 14 whenever you use ICSF panels to update SYM-MK and ASYM-MK in the new master key register in a PCI Cryptographic Coprocessor. Subtype 14 contains the following information:

- The master Key (SYM-MK or ASYM-MK; NMK or key part) valid indicator
- The new master key verification pattern
- The key part verification pattern
- The PCI Cryptographic Coprocessor processor number
- The PCI Cryptographic Coprocessor serial number
- The domain index

PCI Cryptographic Coprocessor Retained Key Create or Delete (Subtype 15)

ICSF writes to subtype 15 whenever you create or delete a retained private key in a PCI Cryptographic Coprocessor. Subtype 15 contains the following information:

- The operation performed (created, deleted from PCI, deleted from PKDS)
- The retained key label
- The PCI Cryptographic Coprocessor processor number
- The PCI Cryptographic Coprocessor serial number
- The domain index

PCI Cryptographic Coprocessor TKE Command Request or Reply (Subtype 16)

ICSF writes to subtype 16 whenever a TKE workstation either issues a command request to a PCI Cryptographic Coprocessor or receives a reply response from a PCI Cryptographic Coprocessor. Subtype 16 contains the following information:

- The indicator for request or reply
- The PCI Cryptographic Coprocessor processor number
- The PCI Cryptographic Coprocessor serial number
- The PCI Cryptographic Coprocessor domain index
- The request command block or reply response block length
- The request command data block or reply response data block length
- The request or reply CPRB

PCI Cryptographic Coprocessor Timing (Subtype 17)

ICSF periodically records processing times for PCI Cryptographic Coprocessor operations in subtype 17. Subtype 17 contains the following information:

- The time immediately before the operation begins
- The time immediately after the operation ends
- The time immediately after the results of the operation have been communicated to the caller address space
- The number of processes waiting to submit work to the same PCI Cryptographic Coprocessor, domain, and reference slot used by this operation
- The function code for this operation
- The PCI Cryptographic Coprocessor processor number
- The PCI Cryptographic Coprocessor serial number
- The PCI Cryptographic Coprocessor domain
- A reference number that identifies an internal ICSF queue element

Cryptographic Coprocessor Configuration (Subtype 18)

ICSF writes subtype 18 when a PCI Cryptographic Coprocessor or PCI Cryptographic Accelerator is brought online or taken offline. Subtype 18 contains the following information:

- The operation performed (coprocessor brought online or taken offline)
- The coprocessor number
- The PCI Cryptographic Coprocessor serial number or coprocessor number for a PCI Cryptographic Accelerator

Message Recording

ICSF writes messages to data sets and consoles. You can view some messages immediately as they appear on the console, and you can view messages in the data sets.

ICSF writes messages for ICSF mainline processing, key generator utility program (KGUP) processing, and conversion program processing to separate data sets. When the ICSF main task has an abnormal end, messages appear in a data set that you define using the CSFLIST DD statement in the ICSF startup procedure. KGUP processing messages appear in the KGUP diagnostic data set that you define using the CSFDIAG DD statement in the job control language to run KGUP. Conversion program messages appear in the conversion program activity report data set that you define using the CSFVRPT DD statement in the job control language to run the conversion program.

ICSF mainline also issues certain messages to the security console and operator console. Messages with a routing code of 9 appear on the security console. Messages with a routing code of 1 appear on the operator console. If a message does not have a routing code, it appears in the CSFLIST data set.

For a description of each ICSF message and its routing code, see *z/OS ICSF Messages*.

Security Considerations

You can provide enhanced security on ICSF by controlling access to resources and changing the values of your keys periodically. This chapter describes the following aspects of security:

- Controlling access to the key generator utility program (KGUP)
- Controlling access to the callable services
- Controlling access to cryptographic keys
- Scheduling changes for cryptographic keys
- Controlling access to panel functions

Controlling the Program Environment

Some programs or applications which use ICSF require that the environment be program controlled. In a program controlled environment, programs within the address space are defined to the Security Server (RACF). Defining a program to RACF requires the program name and the name of the data set which contains the program.

The commands to define the ICSF load module to RACF are:

```
RDEFINE PROGRAM * ADDMEM('CSF.SCSFMOD0'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

Additional detains on program control may be found in the ″Program Control″ section of the *z/OS Security Server RACF Security Administrator's Guide*.

Controlling Access to KGUP

Anyone running the key generator utility program can read and alter an unprotected cryptographic key data set (CKDS). Therefore, only authorized users should have access to the key generator utility program. To make it difficult for an unauthorized person to execute the key generator utility program, store the program in an APF-authorized library that is protected by the Security Server (RACF).

Controlling Access to the Callable Services

Unauthorized persons should not perform the cryptographic or key management functions that the callable services provide. The security administrator should be the only one able to access some services like those used in managing keys. The security administrator can give access to some services, such as enciphering and deciphering data, to persons who are authorized on the system.

You can use the Security Server (RACF) to control which users can use ICSF callable services. For example, you can use the key export service to export any type of key. Your installation may want only the security administrator to be able to use the key export function.

ICSF provides security exit points that you can use to control access to a callable service. (In ICSF/MVS Version 2 Release 1 the IBM-supplied security exit routines were removed, but the exit points still remain.) For information about the security exit points, see “Security Installation Exits” on page 114.

Your installation may want other users to just be able to export data keys, because sending encrypted data between systems is a common function. The data key export callable service permits the export of data keys only. Your security administrator can have access to the key export service and can use the Security Server (RACF) to give other users access to the data key export service. For more information on controlling who can use ICSF callable services, see *z/OS ICSF Administrator's Guide*.

Access control points for specific functions may be enabled/disabled through the TKE workstation. See “Access to Callable Services” on page 52 and the *z/OS ICSF TKE Workstation User's Guide 2000*, SA22-7524 for additional information.

Controlling Access to Cryptographic Keys

Besides the key generator utility program and services, your installation should also control access to the cryptographic keys. First, it is highly recommended that you store cryptographic keys in data sets that are protected by RACF or an equivalent product. You should limit access to authorized persons or applications. Second, you can use RACF to control access to keys in the in-storage cryptographic key data set. For more information on protecting cryptographic keys, see *z/OS ICSF Administrator's Guide*.

ICSF also provides security exit points that you can use to control access to keys in the in-storage CKDS and in the PKDS. For information about the security exit points, see “Security Installation Exits” on page 114.

Scheduling Changes for Cryptographic Keys

You should periodically change the value of cryptographic keys to reduce the possibility of exposing a key value. It is recommended that you change the DES master key at least every 12 months.

The security administrator can use the key generator utility program (KGUP) to change the cryptographic keys. KGUP updates keys in the disk copy of the cryptographic key data set while the callable services access keys in the in-storage copy of the cryptographic key data set. Therefore, you can change the keys without affecting cryptographic operations. For more information on using KGUP, refer to *z/OS ICSF Administrator's Guide*.

Controlling Access to Administrative Panel Functions

You can perform many ICSF administration functions by using the TSO panels. RACF can protect access to these functions. The functions include:

- Refreshing the CKDS
- Setting the master key
- Changing the master key

Additionally, for S/390 Enterprise Servers, S/390 Multiprise servers, and IBM @server zSeries, the following functions are also protected:

- Clear key entry (access can also be controlled through the TKE workstation, domain controls)
- Passphrase MK/CKDS initialization
- User control functions (enabling and disabling dynamic CKDS access, PKA callable services, PKDS read access, and PKDS write, create, and delete access)

These functions are treated the same way as callable services. See *z/OS ICSF Administrator's Guide* for more information.

Debugging Aids

This chapter contains information you can use when diagnosing problems on ICSF. This chapter describes:

- Component trace
- ICSF SVC 143
- Abnormal endings
- Using the IPCS formatting routine

Component Trace

The ICSF component trace is written to a single buffer that is addressed from the ICSF CCVE. This buffer contains the number of entries you specify in the installation options data set TRACEENTRY parameter. If you do not specify this installation option, the default is 1000. Each entry is 96 bytes long. When the buffer is full, the trace is wrapped.

Examining the Trace Entry Buffer

To examine the trace buffer, use the CTRACE facility of the Interactive Problem Control System (IPCS) in either batch or online mode.

CSF TRACE Common Header: The following items are present in every trace entry:

ASCB@ Address of the (application) ASCB
TCB@ Address of the (application) TCB

ASID Application's address space identifier

These items are omitted in the IPCS trace SUMMARY output, and the type-specific data appears after this common header.

Service and Exit Trace Entry Types: ICSF component trace is always active, and the following types of trace entries are always written to the buffer:
BSERVICE: Before the call to service
ASERVICE: After the call to service
BEXIT: Before the call to exit
AEXIT: After the call to exit

Type-Specific Data for Service Trace Entries: The following items are traced for BSERVICE and ASERVICE entries:
Module: Name of the service called
Rcode: Return code from the service
Reason: Reason code from the service

Rcode and Reason are meaningless for BSERVICE.

These items appear in both IPCS SUMMARY and IPCS FULL output, and the exit trace entries are not called.

Instruction Trace Entry Types: You can activate the following instruction trace entries, in addition to the service and exit trace entry information that is always provided:
BCRYPTO: Before the cryptographic instruction
ACRYPTO: After the cryptographic instruction

To activate the ICSF component trace for instruction trace entries, use the following TRACE ON command:

```
TRACE CT,ON,COMP=CSF
```

Follow the TRACE ON command with this reply:

```
R nn,END
```

To deactivate instruction trace entries, use the following TRACE OFF command:

```
TRACE CT,OFF,COMP=CSF
```

Type-Specific Data for Instruction Trace Entries: The following items are traced for BCRYPTO and ACRYPTO entries:
GPRnn: General Registers 0–13
ARnn: Access Registers 3 and 8
Instruction: The cryptographic instruction

These items appear in both IPCS SUMMARY and IPCS FULL output.

The precise meanings of the register differ for each cryptographic instruction. Indeed, the registers GPR10-GPR13 are not used by any cryptographic instruction. However, the more common registers are:

GPR00 Function called (for example, for CMD, encipher or decipher).

GPR01 Cryptographic status (only valid for ACRYPTO).

GPR02 Address of the local instruction parameter block. The length and

usage of the parameter block differ from instruction to instruction and the usage from function to function within the instruction.

GPR03 Address of output text when this is of variable length (for example, ciphertext for encipher command).

GPR08 Address of input text when this is of variable length (for example, plain text for encipher command).

AR03 Access Register 3 (only useful if GPR03 is useful).

AR08 Access Register 8 (only useful if GPR08 is useful).

ICSF System SVC 143

SVC 143 (0A8F) is an ICSF system SVC that is used by CUSP and PCF macros (GENKEY, RETKEY, CIPHER, and EMK) for SVC entry into ICSF. The SVC allows you to run a CUSP or PCF application on ICSF. See “Running PCF and z/OS ICSF on the Same System” on page 33 for more information about running CUSP and PCF applications on ICSF.

SVC 143 is a type 4 SVC and does not get a lock. The General Trace Facility data is:

r141 and R0 No applicable data.

R1 Address of the parameter list. The macro that is called determines the parameter list.

Abnormal Endings

ICSF has an abnormal ending in only the following cases:
- When an error occurs during ICSF initialization
- When you specify FAIL(ICSF) in the callable service exit installation option
- When the setting of a cryptographic domain index fails

If an abnormal end occurs in any other cases, your application or unit of work ends; however, ICSF is still available.

ICSF has an abnormal end code unique to ICSF. Errors specific to ICSF result in an abnormal end code of X'18F' and a unique reason code. In general, all abnormal ends occurring within ICSF result in an appropriate system dump, user dump, or LOGREC recording.

Review the reason code to see if the abnormal end was an installation or user error. For a list of the reason codes for abnormal end code X'18F', refer to *z/OS MVS System Codes*. If you cannot resolve the problem, save the dump and contact the IBM Support Center.

IPCS Formatting Routine

You can use the Interactive Problem Control System (IPCS) to format and display the certain ICSF control blocks. The IPCS CBFORMAT command displays the control block’s eye-catcher name, its location in the address space, and its field names with their offsets. You specify a symbol with the command to identify the control block. Table 18 on page 147 lists the control blocks you can display, the symbol IPCS recognizes for each control block, and a reference for the control block format.

Table 18. IPCS Symbols and Format References for the ICSF Control Blocks

Control Block	Symbol	Format Reference
Installation-defined Service Table	CSFMGST	Varies for each installation.
CSF Exit Name Table	CSFENT	See Table 7 on page 93.
Cryptographic Communication Vector Table	CSFCCVT	See Table 35 on page 168.
Cryptographic Communication Vector Table Extension	CSFCCVE	See Table 36 on page 173.
Secondary Parameter Block	CSFASPB	See Table 11 on page 103.

For example, to format and display the ICSF Exit Name table issue the following command:

```
CBFORMAT CSFENT
```

Instead of using a symbol to identify the control block, you can provide an address. Find and specify the address of the control block in the address space at the time of the dump. When you specify an address, you must also specify the STRUCTURE keyword with the control block symbol.

Note: To format the secondary parameter block, you must provide an address to identify the control block.

For example, if the address of the secondary parameter block is F632D0, issue the following command to format the secondary parameter block.

```
CBFORMAT F632D0. STRUCTURE(CSFASPB)
```

In the example, the secondary parameter block is located at address F632D0 in the address space at the time of the dump. On the command, you must put a period after the address. With this control block, you also specify the structure keyword with the symbol CSFASPB.

For more information about using the CBFORMAT command, see *z/OS MVS IPCS User's Guide*.

Appendix A. Diagnosis Reference Information

This appendix contains Diagnosis, Modification, or Tuning Information.

This appendix contains descriptions of the cryptographic key data set (CKDS), the public key data set (PKDS), PKA key tokens, the Cryptographic Communication Vector Table (CCVT), and Cryptographic Communication Vector Table Extension (CCVE) data areas.

For more information about key tokens, refer to *z/OS ICSF Application Programmer's Guide*.

Cryptographic Key Data Set (CKDS) Format

Programming Interface information

The CKDS includes a header record, data set record, and an internal DES key token record. Tables in the following sections show the format of each of these records.

Format of the CKDS Header Record

Table 19 presents the format of the CKDS header record.

Table 19. Cryptographic Key Data Set Header Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	72	*Constant*	The field is set to binary zeros and is not used for the header record.
72	8	*Creation date*	The date the CKDS was initialized in the format *yyyymmdd*.
80	8	*Creation time*	The initial time the CKDS was created in the format *hhmmssth*.
88	8	*Last update date*	The most recent date the CKDS was updated, in the format *yyyymmdd*.
96	8	*Last update time*	The most recent time the CKDS was updated, in the format *hhmmssth*.
104	2	*Sequence number*	Initially zero in binary. Incremented each time the data set is processed.
106	2	*CKDS header flag bytes*	Flag bytes. **Bit** — **Meaning When Set On** **0** — The master key verification pattern is valid. **1** — The master key authentication pattern is valid. **2–7** — Reserved. **Note:** After the bits are set on, the given values remain constant in ICSF.
108	8	*Master key verification pattern*	The system master key verification pattern.

Table 19. Cryptographic Key Data Set Header Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
116	8	*Master key authentication pattern*	The system master key authentication pattern.
124	72	*Reserved*	The field is set to binary zeros.
196	52	*Installation data*	Installation data associated with the CKDS record, as supplied by an installation exit.
248	4	*Authentication code*	The code generated by the authentication process that ensures that the CKDS record has not been modified since the last update. The authentication code is placed in the CKDS header record when the CKDS is initialized. ICSF verifies the CKDS header record authentication code whenever a CKDS is reenciphered, refreshed, or converted from PCF to ICSF format.

Format of the CKDS Record

Table 20 presents the format of each data set record.

Table 20. Cryptographic Key Data Set Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	64	*Key label*	The key label specified by the KGUP control statement or Clear Key Input panel when the record was created. When using KGUP and the callable services, you can specify the label to identify the record. The key label is the first field of the key index.
64	8	*Key type*	The type of key the record contains. The master key variant for the key type enciphers the key. A KGUP control statement or Clear Key Input panel specifies the key type when the record is created. The key type is the second field of the key index.
72	8	*Creation date*	The initial date the CKDS record was created in the format *yyyymmdd*.
80	8	*Creation time*	The initial time the CKDS record was created in the format *hhmmssth*.
88	8	*Last update date*	The most recent date the CKDS record was updated in the format *yyyymmdd*.
96	8	*Last update time*	The most recent time the CKDS record was updated in the format *hhmmssth*.
104	64	*Key token*	The internal key token. A key token contains the key value. Refer to “Format of the DES Internal Key Token” on page 151 for the format of the internal key token.

Table 20. Cryptographic Key Data Set Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
168	2	*CKDS flag bytes*	Flag bytes. **Bit** — **Meaning When Set On** **0** — The key within the key token field (offset 104) is a partial key. You can enter key parts through the key entry hardware. A partial key is a key whose final key part has not been entered yet. **1** — Cryptographic key token (CKT) delete. **2** — CKDS label must be unique. **3–7** — Reserved. **Note:** When bit 0 is off, the key within the key token field (offset 104) is an entire key.
170	26	*Reserved*	Reserved.
196	52	*Installation data*	Installation data associated with the CKDS record as supplied by an installation exit.
248	4	*Authentication code*	The code generated by the authentication process that ensures the CKDS record has not been modified since the last update. The authentication code is placed in the CKDS record when the record is created. When you refresh, reencipher, or convert a CKDS, ICSF verifies each CKDS record as ICSF performs the action.

Format of the DES Internal Key Token

Table 21 shows the format for a DES internal key token.

Table 21. Internal Key Token Format

Bytes	Description
0	X'01' (flag indicating this is an internal key token)
1–3	Implementation-dependent bytes (X'000000' for ICSF)
4	Key token version number (X'00' or X'01')
5	Reserved (X'00')
6	Flag byte **Bit** — **Meaning When Set On** **0** — Encrypted key and master key verification pattern (MKVP) are present. **1** — Control vector (CV) value in this token has been applied to the key. **2** — Key is used for no control vector (NOCV) processing. Valid for transport keys only. **3** — Key is an ANSI key-encrypting key (AKEK). **4** — AKEK is a double-length key (16 bytes). **Note:** When bit 3 is on and bit 4 is off, AKEK is a single-length key (8 bytes). **5** — AKEK is partially notarized. **6** — Key is an ANSI partial key. **7** — Export prohibited.
7	Reserved (X'00')

Table 21. Internal Key Token Format (continued)

Bytes	Description
8–15	Master key verification pattern (MKVP)
16–23	A single-length key, the left half of a double-length key, or Part A of a triple-length key. The value is encrypted under the master key.
24–31	X'0000000000000000' if a single-length key, or the right half of a double-length operational key, or Part B of a triple-length operational key. The right half of the double-length key or Part B of the triple-length key is encrypted under the master key.
32–39	The control vector (CV) for a single-length key or the left half of the control vector for a double-length key.
40–47	X'0000000000000000' if a single-length key or the right half of the control vector for a double-length operational key.
48–55	X'0000000000000000' if a single-length key or double-length key, or Part C of a triple-length operational key. Part C of a triple-length key is encrypted under the master key.
56-58	Reserved (X'000000')
59 bits 0 and 1	**B'10'** Indicates CDMF DATA or KEK. **B'00'** Indicates DES for DATA keys or the system default algorithm for a KEK. **B'01'** Indicates DES for a KEK.
59 bits 2 and 3	**B'00'** Indicates single-length key (version 0 only). **B'01'** Indicates double-length key (version 1 only). **B'10'** Indicates triple-length key (version 1 only).
59 bits 4 –7	B'0000'
60–63	Token validation value (TVV).

Note: A key token stored in the CKDS will not have an MKVP or TVV. Before such a key token is used, the MKVP is copied from the CKDS header record and the TVV is calculated and placed in the token. See “Token Validation Value” for more information.

Token Validation Value

ICSF uses the *token validation value (TVV)* to verify that a token is valid. The TVV prevents a key token that is not valid or that is overlaid from being accepted by ICSF. It provides a checksum to detect a corruption in the key token.

When an ICSF callable service generates a key token, it generates a TVV and stores the TVV in bytes 60-63 of the key token. When an application program passes a key token to a callable service, ICSF checks the TVV. To generate the TVV, ICSF performs a twos complement ADD operation (ignoring carries and overflow) on the key token, operating on four bytes at a time, starting with bytes 0-3 and ending with bytes 56-59.

End of Programming Interface information

Public Key Data Set (PKDS) Format

The PKDS record includes the PKDS header and the PKA key token. Tables in the following sections show the format of each of these records.

Format of the PKDS Header Record

Table 22. Public Key Data Set Header Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	64	*PKHVKEY*	VSAM key of the PKDS header.
64	8		Reserved.
72	8	*PKHCRDTE*	The date the PKDS was created in the format *yyyymmdd*.
80	8	*PKHCRTIM*	The initial time the PKDS was created in the format *hhmmssth*.
88	8	*PKHUPDTE*	The most recent date the PKDS header was updated, in the format *yyyymmdd*.
96	8	*PKHUPTIM*	The most recent time the PKDS header was updated, in the format *hhmmssth*.
104	4	*PKHRLEN*	Length of the PKDS header entry.
108	16	*PKHKMKHP*	The hash pattern of the KMMK.
124	16	*PKHSMKHP*	The hash pattern of the SMK.
140	20		Reserved.
160	20	*PKHAUTH*	PKDS header authentication code.

Format of the PKDS Record

Table 23. Public Key Data Set Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	64	*PKDLABEL*	Label or name of this PKDS entry.
64	8		Reserved.
72	8	*PKDCRDTE*	The date this PKDS record was created in the format *yyyymmdd*.
80	8	*PKDCRTIM*	The initial time this PKDS record was created in the format *hhmmssth*.
88	8	*PKDUPDTE*	The most recent date this PKDS record was updated, in the format *yyyymmdd*.
96	8	*PKDUPTIM*	The most recent time this PKDS record was updated, in the format *hhmmssth*.
104	4	*PKDRLEN*	Length of the entire PKDS record entry.
108	52	*PKDUDATA*	User data.
160	20	*PKDAUTH*	The entry authentication code.
180	1868	*PKDTOKEN*	The public or private key token.

PKA Token Formats

As with DES key tokens, the first byte of a PKA key token indicates the type of token. If the first byte of the key identifier is X'1E' or X'1F', this indicates that it is a **PKA key token**.

A first byte of X'1E' indicates an external token with a cleartext public key and optionally a private key that is either in cleartext or enciphered by a transport key-encrypting key.

A first byte of X'1F' indicates an internal token with a cleartext public key and a private key that is enciphered by the master key and ready for internal use.

Although DES tokens are 64 bytes, PKA tokens are of variable length because they contain either RSA or DSS key values, which are variable in length. Consequently, length parameters precede all PKA token parameters. The maximum allowed size is 2500 bytes. PKA key tokens consist of a token header, any required sections, and optional sections, which depend on the token type.

A PKA key token can be a public or private key token, and a private key token can be internal or external. Therefore, there are three basic types of tokens, each of which can contain either RSA or DSS information:
- Public key tokens
- Private external key tokens
- Private internal key tokens

Public key tokens contain only the public key. Private key tokens contain the public and private key pair.

This appendix presents formats for:
- An RSA public key token
- A DSS public key token
- An RSA private external key token
- A DSS private external key token
- An RSA private internal key token
- A DSS private internal key token

Format of the RSA Public Key Token

An RSA public key token contains the following sections:
- A required token header, starting with the token identifier X'1E'
- A required RSA public key section, starting with the section identifier X'04'

Table 24 presents the format of an RSA public key token. All length fields are in binary. All binary fields (exponents, lengths, and so on) are stored with the high-order byte first (left, low-address, S/390 format).

Table 24. RSA Public Key Token

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier. X'1E' indicates an external token.
001	001	Version, X'00'.
002	002	Length of the key token structure.
004	004	Ignored. Should be zero.
RSA Public Key Section (required)		
000	001	X'04', section identifier, RSA public key.
001	001	X'00', version.
002	002	Section length, 12+xxx+yyy.
004	002	Reserved field.

Table 24. RSA Public Key Token (continued)

Offset (Dec)	Number of Bytes	Description
006	002	RSA public key exponent field length in bytes, “xxx”.
008	002	Public key modulus length in bits.
010	002	RSA public key modulus field length in bytes, “yyy”.
012	xxx	Public key exponent (this is generally a 1-, 3-, or 64- to 256-byte quantity), e. e must be odd and 1<e<n. (Frequently, the value of e is 2.) ^{16}+1
12+xxx	yyy	Modulus, n.

Format of the DSS Public Key Token

A DSS public key token contains the following sections:

- A required token header, starting with the token identifier X'1E'
- A required DSS public key section, starting with the section identifier X'03'

Table 25 presents the format of a DSS public key token. All length fields are in binary. All binary fields (exponents, lengths, and so on) are stored with the high-order byte first (left, low-address, S/390 format).

Table 25. DSS Public Key Token

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier. X'1E' indicates an external token.
001	001	Version, X'00'.
002	002	Length of the key token structure.
004	004	Ignored. Should be zero.
DSS Public Key Section (required)		
000	001	X'03', section identifier, DSS public key.
001	001	X'00', version.
002	002	Section length, 14+ppp+qqq+ggg+yyy.
004	002	Size of p in bits. The size of p must be one of: 512, 576, 640, 704, 768, 832, 896, 960, or 1024.
006	002	Size of the p field in bytes, “ppp”.
008	002	Size of the q field in bytes, “qqq”.
010	002	Size of the g field in bytes, “ggg”.
012	002	Size of the y field in bytes, “yyy”.
014	ppp	Prime modulus (large public modulus), p.
014 +ppp	qqq	Prime divisor (small public modulus), q. $2^{159}<q<2^{160}$.
014 +ppp +qqq	ggg	Public key generator, g.
014 +ppp +qqq +ggg	yyy	Public key, y. $y=g^x \bmod(p)$; 1<y<p.

Format of RSA Private External Key Tokens

An RSA private external key token contains the following sections:

- A required PKA token header starting with the token identifier X'1E'
- A required RSA private key section starting with one of the following section identifiers:

  - X'02' which indicates a modulus-exponent form RSA private key section (not optimized) with modulus length of up to 1024 bits for use with the Cryptographic Coprocessor Feature or the PCI Cryptographic Coprocessor.
  - X'08' which indicates an optimized Chinese Remainder Theorem form private key section with modulus bit length of up to 2048 bits for use with the PCI Cryptographic Coprocessor
- A required RSA public key section, starting with the section identifier X'04'
- An optional private key name section, starting with the section identifier X'10'

Table 26 presents the basic record format of an RSA private external key token. All length fields are in binary. All binary fields (exponents, lengths, and so on) are stored with the high-order byte first (left, low-address, S/390 format). All binary fields (exponents, modulus, and so on) in the private sections of tokens are right-justified and padded with zeros to the left.

Table 26. RSA Private External Key Token Basic Record Format

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier. X'1E' indicates an external token. The private key is either in cleartext or enciphered with a transport key-encrypting key.
001	001	Version, X'00'.
002	002	Length of the key token structure.
004	004	Ignored. Should be zero.
RSA Private Key Section (required)		
• For 1024-bit Modulus-Exponent form refer to “RSA Private Key Token, 1024-bit Modulus-Exponent External Form” on page 157 • For 2048-bit Chinese Remainder Theorem form refer to “RSA Private Key Token, 2048-bit Chinese Remainder Theorem External Form” on page 157		
RSA Public Key Section (required)		
000	001	X'04', section identifier, RSA public key.
001	001	X'00', version.
002	002	Section length, 12+xxx.
004	002	Reserved field.
006	002	RSA public key exponent field length in bytes, “xxx”.
008	002	Public key modulus length in bits.
010	002	RSA public key modulus field length in bytes, which is zero for a private token. **Note:** In an RSA private key token, this field should be zero. The RSA private key section contains the modulus.
012	xxx	Public key exponent, e (this is generally a 1-, 3-, or 64- to 256-byte quantity). e must be odd and 1<e<n. (Frequently, the value of e is 2^{16}+1 (=65,537).
Private Key Name (optional)		
000	001	X'10', section identifier, private key name.
001	001	X'00', version.
002	002	Section length, X'0044' (68 decimal).
004	064	Private key name (in ASCII), left-justified, padded with space characters (X'20'). An access control system can use the private key name to verify that the calling application is entitled to use the key.

RSA Private Key Token, 1024-bit Modulus-Exponent External Form: This RSA private key token and the external X'02' token is supported on the Cryptographic Coprocessor Feature and PCI Cryptographic Coprocessor.

Table 27. RSA Private Key Token, 1024-bit Modulus-Exponent External Format

Offset (Dec)	Number of Bytes	Description
000	001	X'02', section identifier, RSA private key, modulus-exponent format (RSA-PRIV)
001	001	X'00', version.
002	002	Length of the RSA private key section X'016C' (364 decimal).
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the section end. This hash value is checked after an enciphered private key is deciphered for use.
024	004	Reserved; set to binary zero.
028	001	Key format and security: **X'00'** Unencrypted RSA private key subsection identifier. **X'82'** Encrypted RSA private key subsection identifier.
029	001	Reserved, binary zero.
030	020	SHA-1 hash of the optional key-name section. If there is no key-name section, then 20 bytes of X'00'.
050	004	Key use flag bits. **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. All other bits reserved, set to binary zero.
054	006	Reserved; set to binary zero.
060	024	Reserved; set to binary zero.
084	Start of the optionally-encrypted secure subsection.	
084	024	Random number, confounder.
108	128	Private-key exponent, d. $d=e^{-1} \bmod((p-1)(q-1))$, and $1<d<n$ where e is the public exponent.
	End of the optionally-encrypted subsection; the confounder field and the private-key exponent field are enciphered for key confidentiality when the key format and security flags (offset 28) indicate that the private key is enciphered. They are enciphered under a double-length transport key using the ede2 algorithm.	
236	128	Modulus, n. n=pq where p and q are prime and $1<n<2^{1024}$.

RSA Private Key Token, 2048-bit Chinese Remainder Theorem External Form: This RSA private key token is supported on the PCI Cryptographic Coprocessor.

Table 28. RSA Private Key Token, 2048-bit Chinese Remainder Theorem External Format

Offset (Dec)	Number of Bytes	Description
000	001	X'08', section identifier, RSA private key, CRT format (RSA-CRT)
001	001	X'00', version.
002	002	Length of the RSA private-key section, 132 + ppp + qqq + rrr + sss + uuu + xxx + nnn.

Table 28. RSA Private Key Token, 2048-bit Chinese Remainder Theorem External Format (continued)

Offset (Dec)	Number of Bytes	Description
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the end of the modulus.
024	004	Reserved; set to binary zero.
028	001	Key format and security: **X'40'** Unencrypted RSA private-key subsection identifier, Chinese Remainder form. **X'42'** Encrypted RSA private-key subsection identifier, Chinese Remainder form.
029	001	Reserved; set to binary zero.
030	020	SHA-1 hash of the optional key-name section and any following optional sections. If there are no optional sections, then 20 bytes of X'00'.
050	004	Key use flag bits. **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. All other bits reserved, set to binary zero.
054	002	Length of prime number, p, in bytes: ppp.
056	002	Length of prime number, q, in bytes: qqq.
058	002	Length of d_p, in bytes: rrr.
060	002	Length of d_q, in bytes: sss.
062	002	Length of U, in bytes: uuu.
064	002	Length of modulus, n, in bytes: nnn.
066	004	Reserved; set to binary zero.
070	002	Length of padding field, in bytes: xxx.
072	004	Reserved, set to binary zero.
076	016	Reserved, set to binary zero.
092	032	Reserved; set to binary zero.
124	Start of the optionally-encrypted secure subsection.	
124	008	Random number, confounder.
132	ppp	Prime number, p.
132 + ppp	qqq	Prime number, q
132 + ppp + qqq	rrr	$d_p = d \bmod(p - 1)$
132 + ppp + qqq + rrr	sss	$d_q = d \bmod(q - 1)$
132 + ppp + qqq + rrr + sss	uuu	$U = q^{-1} \bmod(p)$.
132 + ppp + qqq + rrr + sss + uuu	xxx	X'00' padding of length xxx bytes such that the length from the start of the random number above to the end of the padding field is a multiple of eight bytes.
	End of the optionally-encrypted secure subsection; all of the fields starting with the confounder field and ending with the variable length pad field are enciphered for key confidentiality when the key format-and-security flags (offset 28) indicate that the private key is enciphered. They are enciphered under a double-length transport key using the TDES (CBC outer chaining) algorithm.	

Table 28. RSA Private Key Token, 2048-bit Chinese Remainder Theorem External Format (continued)

Offset (Dec)	Number of Bytes	Description
132 + ppp + qqq + rrr + sss + uuu + xxx	nnn	Modulus, n. n = pq where p and q are prime and $2^{512}<n<2^{2048}$.

Format of the DSS Private External Key Token

A DSS private external key token contains the following sections:

- A required PKA token header, starting with the token identifier X'1E'
- A required DSS private key section, starting with the section identifier X'01'
- A required DSS public key section, starting with the section identifier X'03'
- An optional private key name section, starting with the section identifier X'10'

Table 29 presents the format of a DSS private external key token. All length fields are in binary. All binary fields (exponents, lengths, and so on) are stored with the high-order byte first (left, low-address, S/390 format). All binary fields (exponents, modulus, and so on) in the private sections of tokens are right-justified and padded with zeros to the left.

Table 29. DSS Private External Key Token

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier. X'1E' indicates an external token. The private key is enciphered with a PKA master key.
001	001	Version, X'00'.
002	002	Length of the key token structure.
004	004	Ignored. Should be zero.
DSS Private Key Section and Secured Subsection (required)		
000	001	X'01', section identifier, DSS private key.
001	001	X'00', version.
002	002	Length of the DSS private key section, 436, X'01B4'.
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the section end. This hash value is checked after an enciphered private key is deciphered for use.
024	004	Reserved; set to binary zero.
028	001	Key security: **X'00'** Unencrypted DSS private key subsection identifier. **X'81'** Encrypted DSS private key subsection identifier.
029	001	Padding, X'00'.
030	020	SHA-1 hash of the key token structure contents that follow the public key section. If no sections follow, this field is set to binary zeros.
050	010	Reserved; set to binary zero.
060	048	Ignored; set to binary zero.
108	128	Public key generator, g. 1<g<p.
236	128	Prime modulus (large public modulus), p. $2^{L-1}<p<2^{L}$ and L (the modulus length) must be a multiple of 64.
364	020	Prime divisor (small public modulus), q. $2^{159}<q<2^{160}$.
384	004	Reserved; set to binary zero.

Table 29. DSS Private External Key Token (continued)

Offset (Dec)	Number of Bytes	Description
388	024	Random number, confounder. **Note:** This field and the next two fields are enciphered for key confidentiality when the key security flag (offset 28) indicates the private key is enciphered.
412	020	Secret DSS key, x; x is random. (See the preceding note.)
432	004	Random number, generated when the secret key is generated. (See the preceding note.)
DSS Public Key Section (required)		
000	001	X'03', section identifier, DSS public key.
001	001	X'00', version.
002	002	Section length, 14+yyy.
004	002	Size of p in bits. The size of p must be one of: 512, 576, 640, 704, 768, 832, 896, 960, or 1024.
006	002	Size of the p field in bytes, which is zero for a private token.
008	002	Size of the q field in bytes, which is zero for a private token.
010	002	Size of the g field in bytes, which is zero for a private token.
012	002	Size of the y field in bytes, “yyy”.
014	yyy	Public key, y. $y=g^x \bmod(p)$ **Note:** p, q, and y are defined in the DSS public key token.
Private Key Name (optional)		
000	001	X'10', section identifier, private key. name
001	001	X'00', version.
002	002	Section length, X'0044' (68 decimal).
004	064	Private key name (in ASCII), left-justified, padded with space characters (X'20'). An access control system can use the private key name to verify that the calling application is entitled to use the key.

Internal PKA Tokens

Programming Interface information

PKA private internal key tokens contain both private and public key information. There is no need for an internal token with only the public key information because the public values are in the clear.

The first byte of X'1F' indicates an internal token with a cleartext public key and a private key that is enciphered with a PKA master key and ready for local (internal) use.

The format of a PKA private internal key token is similar to that of a private external token. The only differences are changes in the private key section and the addition of some internal information at the end of the token. This last section starts with the eyecatcher 'PKTN' rather than with a token or section marker.

Format of the RSA Private Internal Key Token

An RSA private internal key token contains the following sections:

- A required PKA token header, starting with the token identifier X'1F'

- basic record format of an RSA private internal key token. All length fields are in binary. All binary fields (exponents, lengths, and so on) are stored with the high-order byte first (left, low-address, S/390 format). All binary fields (exponents, modulus, and so on) in the private sections of tokens are right-justified and padded with zeros to the left.

Table 30. RSA Private Internal Key Token Basic Record Format

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier. X'1F' indicates an internal token. The private key is enciphered with a PKA master key.
001	001	Version, X'00'.
002	002	Length of the key token structure excluding the internal information section.
004	004	Ignored; should be zero.
RSA Private Key Section and Secured Subsection (required)		
• For 1024-bit X'02' Modulus-Exponent form refer to “RSA Private Key Token, 1024-bit Modulus-Exponent Internal Form for Cryptographic Coprocessor Feature” on page 162 • For 1024-bit X'06' Modulus-Exponent form refer to “RSA Private Key Token, 1024-bit Modulus-Exponent Internal Form for PCI Cryptographic Coprocessor” on page 163 • For 2048-bit X'08' Chinese Remainder Theorem form refer to “RSA Private Key Token, 2048-bit Chinese Remainder Theorem Internal Form” on page 164		
RSA Public Key Section (required)		
000	001	X'04', section identifier, RSA public key.
001	001	X'00', version.
002	002	Section length, 12+xxx.
004	002	Reserved field.
006	002	RSA public key exponent field length in bytes, “xxx”.
008	002	Public key modulus length in bits.
010	002	RSA public key modulus field length in bytes, which is zero for a private token.
012	xxx	Public key exponent (this is generally a 1, 3, or 64 to 256-byte quantity), e. e must be odd and 1<e<n. (Frequently, the value of e is 2^{16}+1 (=65,537).
Private Key Name (optional)		
000	001	X'10', section identifier, private key name.
001	001	X'00', version.
002	002	Section length, X'0044' (68 decimal).
004	064	Private key name (in ASCII), left-justified, padded with space characters (X'20'). An access control system can use the private key name to verify that the calling application is entitled to use the key.
Internal Information Section (required)		
000	004	Eye catcher 'PKTN'.

Table 30. RSA Private Internal Key Token Basic Record Format (continued)

Offset (Dec)	Number of Bytes	Description
004	004	PKA token type. **Bit** **Meaning When Set On** **0** RSA key. **1** DSS key. **2** Private key. **3** Public key. **4** Private key name section exists. **5** Private key unenciphered. **6** Blinding information present. **7** Retained private key.
008	004	Address of token header.
012	002	Total length of total structure including this information section.
014	002	Count of number of sections.
016	016	PKA master key hash pattern.
032	001	Domain of retained key.
033	008	Serial number of processor holding retained key.
041	007	Reserved.

RSA Private Key Token, 1024-bit Modulus-Exponent Internal Form for Cryptographic Coprocessor Feature:

Table 31. RSA Private Internal Key Token, 1024-bit ME Form for Cryptographic Coprocessor Feature

Offset (Dec)	Number of Bytes	Description
000	001	X'02', section identifier, RSA private key.
001	001	X'00', version.
002	002	Length of the RSA private key section X'016C' (364 decimal).
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the section end. This hash value is checked after an enciphered private key is deciphered for use.
024	004	Reserved; set to binary zero.
028	001	Key format and security: **X'02'** RSA private key.
029	001	Format of external key from which this token was derived: **X'21'** External private key was specified in the clear. **X'22'** External private key was encrypted.
030	020	SHA-1 hash of the key token structure contents that follow the public key section. If no sections follow, this field is set to binary zeros.
050	001	Key use flag bits. **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. All other bits reserved, set to binary zero.

Table 31. RSA Private Internal Key Token, 1024-bit ME Form for Cryptographic Coprocessor Feature (continued)

Offset (Dec)	Number of Bytes	Description
051	009	Reserved; set to binary zero.
060	048	Object Protection Key (OPK) encrypted under a PKA master key—can be under the Signature Master Key (SMK) or Key Management Master Key (KMMK) depending on key use.
108	128	Secret key exponent d, encrypted under the OPK. $d=e^{-1} \bmod((p-1)(q-1))$
236	128	Modulus, n. n=pq where p and q are prime and $1<n<2^{1024}$.

RSA Private Key Token, 1024-bit Modulus-Exponent Internal Form for PCI Cryptographic Coprocessor:

Table 32. RSA Private Internal Key Token, 1024-bit ME Form for PCI Cryptographic Coprocessor

Offset (Dec)	Number of Bytes	Description
000	001	X'06', section identifier, RSA private key modulus-exponent format (RSA-PRIV).
001	001	X'00', version.
002	002	Length of the RSA private key section X'0198' (408 decimal) + rrr + iii + xxx.
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to and including the modulus at offset 236.
024	004	Reserved; set to binary zero.
028	001	Key format and security: **X'02'** RSA private key.
029	001	Format of external key from which this token was derived: **X'21'** External private key was specified in the clear. **X'22'** External private key was encrypted. **X'23'** Private key was generated using regeneration data. **X'24'** Private key was randomly generated.
030	020	SHA-1 hash of the optional key-name section and any following optional sections. If there are no optional sections, this field is set to binary zeros.
050	004	Key use flag bits. **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. All other bits reserved, set to binary zeros.
054	006	Reserved; set to binary zero.
060	048	Object Protection Key (OPK) encrypted under the Asymmetric Keys Master Key using the ede3 algorithm.
108	128	Private key exponent d, encrypted under the OPK using the ede5 algorithm. $d=e^{-1}\bmod((p-1)(q-1))$, and $1<d<n$ where e is the public exponent.
236	128	Modulus, n. n=pq where p and q are prime and $2^{512}<n<2^{1024}$.
364	016	Asymmetric-Keys Master Key hash pattern.
380	020	SHA-1 hash value of the blinding information subsection cleartext, offset 400 to the end of the section.
400	002	Length of the random number r, in bytes: rrr.

Table 32. RSA Private Internal Key Token, 1024-bit ME Form for PCI Cryptographic Coprocessor (continued)

Offset (Dec)	Number of Bytes	Description
402	002	Length of the random number r^{-1}, in bytes: iii.
404	002	Length of the padding field, in bytes: xxx.
406	002	Reserved; set to binary zeros.
408	Start of the encrypted blinding subsection	
408	rrr	Random number r (used in blinding).
408 + rrr	iii	Random number r^{-1} (used in blinding).
408 + rrr + iii	xxx	X'00' padding of length xxx bytes such that the length from the start of the encrypted blinding subsection to the end of the padding field is a multiple of eight bytes.
	End of the encrypted blinding subsection; all of the fields starting with the random number r and ending with the variable length pad field are encrypted under the OPK using TDES (CBC outer chaining) algorithm.	

RSA Private Key Token, 2048-bit Chinese Remainder Theorem Internal Form:
This RSA private key token is supported on the PCI Cryptographic Coprocessor.

Table 33. RSA Private Internal Key Token, 2048-bit Chinese Remainder Theorem External Format

Offset (Dec)	Number of Bytes	Description
000	001	X'08', section identifier, RSA private key, CRT format (RSA-CRT)
001	001	X'00', version.
002	002	Length of the RSA private-key section, 132 + ppp + qqq + rrr + sss + uuu + +ttt + iii + xxx + nnn.
004	020	SHA-1 hash value of the private-key subsection cleartext, offset 28 to the end of the modulus.
024	004	Reserved; set to binary zero.
028	001	Key format and security: **X'08'** Encrypted RSA private-key subsection identifier, Chinese Remainder form.
029	001	Key derivation method: **X'21'** External private key was specified in the clear. **X'22'** External private key was encrypted. **X'23'** Private key was generated using regeneration data. **X'24'** Private key was randomly generated.
030	020	SHA-1 hash of the optional key-name section and any following sections. If there are no optional sections, then 20 bytes of X'00'.
050	004	Key use flag bits: **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. All other bits reserved, set to binary zero.
054	002	Length of prime number, p, in bytes: ppp.
056	002	Length of prime number, q, in bytes: qqq.
058	002	Length of d_p, in bytes: rrr.
060	002	Length of d_q, in bytes: sss.

Table 33. RSA Private Internal Key Token, 2048-bit Chinese Remainder Theorem External Format (continued)

Offset (Dec)	Number of Bytes	Description
062	002	Length of U, in bytes: uuu.
064	002	Length of modulus, n, in bytes: nnn.
066	002	Length of the random number r, in bytes: ttt.
068	002	Length of the random number r^{-1}, in bytes: iii.
070	002	Length of padding field, in bytes: xxx.
072	004	Reserved, set to binary zero.
076	016	Asymmetric-Keys Master Key hash pattern.
092	032	Object Protection Key (OPK) encrypted under the Asymmetric-Keys Master Key using the TDES (CBC outer chaining) algorithm.
124	Start of the encrypted secure subsection, encrypted under the OPK using TDES (CBC outer chaining).	
124	008	Random number, confounder.
132	ppp	Prime number, p.
132 + ppp	qqq	Prime number, q
132 + ppp + qqq	rrr	d_p = d mod(p - 1)
132 + ppp + qqq + rrr	sss	d_q = d mod(q - 1)
132 + ppp + qqq + rrr + sss	uuu	U = q^{-1}mod(p).
132 + ppp + qqq + rrr + sss + uuu	ttt	Random number r (used in blinding).
132 + ppp + qqq + rrr + sss + uuu + ttt	iii	Random number r^{-1} (used in blinding).
132 + ppp + qqq + rrr + sss + uuu + ttt + iii	xxx	X'00' padding of length xxx bytes such that the length from the start of the confounder at offset 124 to the end of the padding field is a multiple of eight bytes.
	End of the encrypted secure subsection; all of the fields starting with the confounder field and ending with the variable length pad field are encrypted under the OPK using TDES (CBC outer chaining) for key confidentiality.	
132 + ppp + qqq + rrr + sss + uuu + ttt + iii + xxx	nnn	Modulus, n. n = pq where p and q are prime and 2^{512}<n<2^{2048}.

Format of the DSS Private Internal Key Token

A DSS private internal key token contains the following sections:

- A required PKA token header, starting with the token identifier X'1F'
- A required DSS private key section, starting with the section identifier X'01'
- A required DSS public key section, starting with the section identifier X'03'
- An optional private key name section, starting with the section identifier X'10'
- A required internal information section, starting with the eyecatcher 'PKTN'

Table 34 on page 166 presents the format of a DSS private internal token. All length fields are in binary. All binary fields (exponents, lengths, and so on) are stored with the high-order byte first (left, low-address, S/390 format). All binary fields (exponents, modulus, and so on) in the private sections of tokens are right-justified and padded with zeros to the left.

Table 34. DSS Private Internal Key Token

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier. X'1F' indicates an internal token. The private key is enciphered with a PKA master key.
001	001	Version, X'00'.
002	002	Length of the key token structure excluding the internal information section.
004	004	Ignored; should be zero.
DSS Private Key Section and Secured Subsection (required)		
000	001	X'01', section identifier, DSS private key.
001	001	X'00', version.
002	002	Length of the DSS private key section, 436, X'01B4'.
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the section end. This hash value is checked after an enciphered private key is deciphered for use.
024	004	Reserved; set to binary zero.
028	001	Key security: X'01' DSS private key.
029	001	Format of external key token: **X'10'** Private key generated on an ICSF host. **X'11'** External private key was specified in the clear. **X'12'** External private key was encrypted.
030	020	SHA-1 hash of the key token structure contents that follow the public key section. If no sections follow, this field is set to binary zeros.
050	010	Reserved; set to binary zero.
060	048	The OPK encrypted under a PKA master key (Signature Master Key (SMK)).
108	128	Public key generator, g. $1<g<p$.
236	128	Prime modulus (large public modulus), p. $2^{L-1}<p<2^{L}$ for $512\leq L\leq 1024$, and L (the modulus length) must be a multiple of 64.
364	020	Prime divisor (small public modulus), q. $2^{159}<q<2^{160}$.
384	004	Reserved; set to binary zero.
388	024	Random number, confounder. **Note:** This field and the two that follow are enciphered under the OPK.
412	020	Secret DSS key, x. x is random. (See the preceding note.)
432	004	Random number, generated when the secret key is generated. (See the preceding note.)
DSS Public Key Section (required)		
000	001	X'03', section identifier, DSS public key.
001	001	X'00', version.
002	002	Section length, 14+yyy.
004	002	Size of p in bits. The size of p must be one of: 512, 576, 640, 704, 768, 832, 896, 960, or 1024.
006	002	Size of the p field in bytes, which is zero for a private token.
008	002	Size of the q field in bytes, which is zero for a private token.
010	002	Size of the g field in bytes, which is zero for a private token.

Table 34. DSS Private Internal Key Token (continued)

Offset (Dec)	Number of Bytes	Description
012	002	Size of the y field in bytes, “yyy”.
014	yyy	Public key, y. $y=g^x \bmod(p)$; **Note:** p, g, and y are defined in the DSS public key token.
Private Key Name (optional)		
000	001	X'10', section identifier, private key name.
001	001	X'00', version.
002	002	Section length, X'0044' (68 decimal).
004	064	Private key name (in ASCII), left-justified, padded with space characters (X'20'). An access control system can use the private key name to verify that the calling application is entitled to use the key.
Internal Information Section (required)		
000	004	Eye catcher 'PKTN'.
004	004	PKA token type. **Bit** **Meaning When Set On** **0** RSA key. **1** DSS key. **2** Private key. **3** Public key. **4** Private key name section exists.
008	004	Address of token header.
012	002	Length of internal work area.
014	002	Count of number of sections.
016	016	PKA master key hash pattern.
032	016	Reserved.

End of Programming Interface information

Data Areas

The following sections present the format of the Cryptographic Communication Vector Table (CCVT) and the Cryptographic Communication Vector Table Extension (CCVE) data areas.

The Cryptographic Communication Vector Table (CCVT)

The CCVT is the ICSF base control block and contains addresses of common areas for use by ICSF components. Indicators in the CCVT also provide ICSF status information. The CCVT is getmained in subpool 245 below the line.

Note: The CCVT is not freemained when you stop ICSF.

Programming Interface information

CCVT

ONLY the following fields are part of the programming interface:

- CCVTDACC
- CCVTCCVE
- CCVTHFLG
- CCVTPRPC
- CCVTINST
- CCVTINS2
- CCVTLNTH
- CCVT_FMID
- CCVT_USERPARM

End of Programming Interface information

Table 35 describes the contents of the Cryptographic Communication Vector Table.

Table 35. Cryptographic Communication Vector Table

Offset (Dec)	Number of Bytes	Field Name	Description
0	4	CCVTID	EBCDIC Cryptographic Communication Vector Table ID. This field must contain the character string `CCVT`.
4	2	CCVTVER	Version. The version of the CCVT. This field must contain the character string `02`.
6	2	CCVTLEN	Length. The length of the CCVT. The value of this field is 296 in decimal.
8	1	CCVTAUX	Auxilliary flags. **Bit** **Meaning When Set On** **0** ICSF is terminating.
9	5		Reserved.
14	2	CCVTRLVL	ICSF level.
16	4	CCVTCCVE	Cryptographic Communication Vector Table Extension (CCVE) address. The address of a private area extension of the CCVT. You should place any fields not needed by other address spaces in the CCVE.
20	4	CCVTPC1	PC number for entry into module CSFANSPC.
24	4	CCVTPC2	PC number for entry into module CSFASSPC.
28	4	CCVTPRPC	Entry point for the pre-PC processing module, CSFARPC.
32	4	CCVTINST	For installation use.

Table 35. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description	
36	1	CCVTSFG1	Status byte.	
			Bit	**Meaning When Set On**
			0	ICSF services are active.
			1	At least one Integrated Cryptographic Feature has a valid master key.
			2	ICSF initialization complete.
			3	ICSF is active and PCF is not active.
			4	Compatibility is permitted. COMPAT(YES) or COMPAT(COEXIST) is specified.
			5	At least one Integrated Cryptographic Feature is valid.
			6	SEC 250 or above.
			7	S/390 Enterprise Servers and S/390 Multiprise Cryptographic Coprocessor Feature is in use.
37	1	CCVTFLAG	Flag byte.	
			Bit	**Meaning When Set On**
			0	VTAM puts terminal buffer above 16MB line.
			1	Hardware environment tested.
			2	GTMAC opcode in hardware.
			3	PCI Cryptographic Coprocessor hardware instructions available.
			4	At least one PCI Cryptographic Coprocessor is active.
			5	Additional hardware environment tested.
			6	At least one PCI Cryptographic Coprocessor is online.
			7	At least one PCI Cryptographic Coprocessor is present.

Table 35. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
38	1	CCVTOFLG	Operational flag byte. **Bit** — **Meaning When Set On** **0** — Configuration is under PR/SM. **1** — Close the CKDS **2** — Key record create, key record delete, and key record write disallowed. **3** — I/O subtask is available. **4** — CCVT_DEF_ALG bit. If on, CDMF is the system default algorithm; if off, DES is the default. **5** — CCVT_CDMF_ENA bit. If on, hardware is capable of performing CDMF. **6** — PKA master keys are valid. **7** — Use ICSF reason codes.
39	1	CCVTSVCM	SVC number for key management. This is the PCF compatibility SVC.
40	1	CCVTCDX	Old CDX, constant this IPL.
41	1	CCVTSVCS	SVC number for DES interface SVC. This is the PCF compatibility SVC.
42	2	CCVTASID	ASID of ICSF address space.
44	4	CCVTIDNR	Subtask caller ID.
48	4	CCVTPC3	Entry point to CSFASSPA used by compatibility SVCs.
52	4	CCVTSRUT	Address of the access method module.
56	8	CCVTINS2	An 8-byte area for installation use.
64	4	CCVTMDS	Data space server PC. PC number for entry to data space server that adds and deletes the in-storage CKDS.
68	4	CCVTLNTH	Maximum installation data length.
72	4	CCVTASCB	ICSF ASCB address.
76	4	CCVTWLST	Address of CICS Wait List.
80	1	CCVTHFLG	Flag bytes. **Bit** — **Meaning When Set On** **0** — Crypto assist instructions available. **1** — Additional secure Crypto device available. **2** — Support for 64-bit callers. **3-7** — Reserved.
81	3		Reserved.
84	4	CCVTENF	ECB for ENF listen.
88	4	CCVTTCB	ICSF maintask TCB address.

Table 35. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
92	4	CCVTTRC	ECB for component trace.
96	4	CCVTECBA	Address of CAMQ ECB array.
100	4	CCVTENF1	Token for ENF listen exit.
104	4	CCVTENF2	Token for ENF listen exit.
108	4	CCVTLX	LX of ICSF address space.
112	8	CCVTDS	Bytes 0–3: Address of the beginning of the current data space. Bytes 4–7: ALET (Access list entry token) of the current data space.
120	4	CCVTLFDE	ECB to post to start the task to search for disabled Integrated Cryptographic Features.
124	4	CCVTIOSE	ECB to post to use I/O subtask.
128	4	CCVTIOSC	ECB posted by I/O subtask.
132	4	CCVTIOAS	ASCB address for non-CSF address space.
136	8	CCVTFMID	ICSF FMID.
144	8	CCVT_USERPARM	ICSF user parameter.
152	1	CCVTPKAF	PKA register clear key entry processing flags. **Bit** — **Meaning When Set On** **0** — KMMK is valid for CP0. **1** — SMK is valid for CP0. **2** — KMMK has been reset for CP0. **3** — SMK has been reset for CP0. **4** — KMMK is valid for CP1. **5** — SMK is valid for CP1. **6** — KMMK has been reset for CP1. **7** — SMK has been reset for CP1.
153	1	CCVTPKAR	**Bit** — **Meaning When Set On** **0 and 1** — SMK status for KSU0. **2 and 3** — KMMK status for KSU0. **4 and 5** — SMK status for KSU1. **6 and 7** — KMMK status for KSU1.
154	1	CCVTPKAX	PKA register status (reserved).
155	1	CCVTKAZ	PKA register status (reserved).
156	16	CCVTCCC	Cryptographic configuration control (CCC).
172	4	CCVTSPKB	Address of public key build.
176	4	CCVTSPKX	Address of public key extract.
180	4	CCVTPIOE	ECB for PKDS I/O subtask.

Table 35. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
184	4	CCVTPIOC	ECB for PKDS I/O work complete.
188	4	CCVTPIOA	Address of ASCB task posting the PKDS I/O subtask.
192	4	CCVTIDNR_PKDS	I/O subtask caller identification.
196	1	CCVTPKDF	PKDS processing flags. **Bit** **Meaning When Set On** **0** PKDS available. **1** Signal PKDS to I/O close PKDS. **2** At least one PCICA is active. **3-7** Reserved.
197	1	CCVTCICS	CICS processing flags. **Bit** **Meaning When Set On** **0** CSFAPRPD installed. **1** CSFACKWL installed.
198	1	CCVTYAFF	**Bit** **Meaning When Set On** **0** ZKA compliance environment.
199	1	*	Reserved.
200	4	CCVTPRPD	Address of CSFAPRPD.
204	4	CCVTCKWL	Address of CSFACKWL.
208	4	CCVTPC4	PC4 (CSFMCAMP) number.
212	4	*	Reserved.
216	4	CCVTENF3	ENF token for PCI Cryptographic Coprocessor online event.
220	4	CCVTENFP	ECB for PCI Cryptographic Coprocessor online event.
224	4	CCVTENA1	Address of ENF1 listen exit.
228	4	CCVTENA2	Address of ENF1 listen exit.
232	4	CCVTENA3	Address of ENF1 listen exit.
236	4	CCVTPC5	PC5 (CSFMCCPP entry).
240	4	CCVTPC6	PC6 (CSFMWCFS entry).
244	16	*	Reserved.
260	4	CCVTGSVT	Address of generic service vector.
264	4	CCVTGSFL	Flags.
268	4	CCVTCSVG	Address of CSFSCVG.
272	4	CCVTACVG	Address of CSFACVG.
276	4	CCVTDACC	ICSF DAC instructions control block for RMF.

Table 35. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
280	1	CCVT_KMC_EXPORT	Hardware feature status. **Bit** **Meaning When Set On** **0** Reserved. **1** KMC DES enabled. **2** Reserved. **3** KMC TDES enabled. **4-7** Reserved.
281	48	*	Reserved.
296		*	Ensure CCVT ends on doubleword boundary.

The Cryptographic Communication Vector Table Extension (CCVE)

The CCVE is an extension of the CCVT that contains fields that can exist. The CCVE exists in ICSF extended private. It should contain any ICSF base control block fields that are not needed by other address spaces.

Programming Interface information

CCVE

ONLY the following fields are part of the programming interface:
- CCVEINPP
- CCVEINPL
- CCVESECC

End of Programming Interface information

Table 36 describes the contents of the Cryptographic Communication Vector Table Extension.

Table 36. Cryptographic Communication Vector Table Extension

Offset (Dec)	Number of Bytes	Field Name	Description
0	4	CCVEID	Cryptographic Communication Vector Table Extension ID. This field must contain the character string `CCVE`.
4	2	CCVEVER	Version. The version number of the CCVE. This field must contain the character string `02`.
6	2	CCVELEN	Length. The length of the CCVE. The value of this field is 368 in decimal.
8	8		Reserved.

Table 36. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
16	4	CCVESTAT	Status word First status byte – CCVESTA1 **Bit** — **Meaning When Set On** **0** — Special secure mode allowed. **1** — Special secure mode enabled. **2** — ICSF is in test mode. **3** — Authentication required for key retrieval. **4** — The hardware has gone from active to inactive. **5** — First start of ICSF during this IPL. **6** — Security Server (RACF) checking required for authorized callers. **7** — PCF coexistence. Second status byte – CCVESTA2 **Bit** — **Meaning When Set On** **0** — Dynamic CKDS updates disallowed. **1** — Refresh needed. **2** — Dynamic CKDS creates disallowed. **3** — Linear CKDS 80% full. **4** — 80% message already sent. **5** — CDMF used (rather than DES). This indicates setting of COMPENC keyword. **6** — PKA callable services disallowed. **7** — Reserved. Third status byte – CCVESTA3 **Bit** — **Meaning When Set On** **0** — PKDS read not permitted. **1** — PKDS write, create, and delete not permitted. **2-7** — Reserved. Remaining 1 byte is reserved.
20	4	CCVECAMQ	Pointer to MCAMQ.
24	4	CCVEEXIT	Pointer to the installation exit router (CSFEXIT).
28	4	CCVESMIB	Address of the storage manager's storage pool. The offset of this field cannot be changed without changing the CSFAGET and CSFAFREE macros.
32	4	CCVETRCE	Address of the *create trace entry* routine.

Table 36. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
36	4	CCVETRCB	Pointer to the current trace buffer. **Bit** **Meaning When Set On** **0** Trace is active.
40	4	CCVECPRM	Address of CPRM.
44	4	CCVEMGST	Address of the generic service table. See “Generic Service Table (CSFMGST)” on page 177 for a description of the generic service table.
48	4	CCVEMQUE	Address of the message queue.
52	4	CCVEENT	Address of the exit name table.
56	4	CCVECC3	Address of the CSFACC3 routine.
60	4	CCVEWM01	Address of CSFWM001.
64	4	CCVEMP01	Address of CSFMP001.
68	4	CCVEMKVN	Master key version numbers. Byte 1: Current master key version number. Bytes 2 and 3: Reserved. Byte 4: Cryptographic domain index.
72	44	CCVECKDS	Data set name of current CKDS.
116	4		Reserved.
120	4		Reserved.
124	4	CCVELFDD	ECB for look for disabled Cryptographic Coprocessor Feature task termination (LFD Done).
128	4	CCVELFDT	Pointer to the TCB for CSFMLFDT.
132	4	CCVECWMB	Address of the common write message block.
136	4	CCVECC3M	Sixth entry to ACC3.
140	4	CCVEFIXS	Address of the fixed area storage used as dynamic storage for the RISGNL routines.
144	4	CCVEFIXL	Length of the fixed area storage.
148	4	CCVECPUF	CPUF routine — used to manipulate the control register.
152	4	CCVERFMK	RFOMK routine — used to RFOMK keys on specific CPs.
156	4	CCVERMKV	MKV RISGNL routine — used by MKV to validate a CP.
160	4	CCVESTHW	STHW routine — used to obtain the current status of the hardware.
164	4	CCVEKEYM	KEYM routine — used to manipulate keys from the key entry hardware.
168	4	CCVEDKEF	DKEF routine — used to manipulate keys for clear key entry.
172	4		Reserved.
176	4		Reserved.
180	4		Reserved.
184	4	CCVECKDL	Pointer to the CKDS lookup routine.

Table 36. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
188	4	CCVECC3A	Pointer to CSFACC3A routine.
192	4	CCVECC3B	Pointer to CSFACC3B routine.
196	4	CCVECC3C	Fourth entry to ACC3.
200	4	CCVECC3L	Fifth entry to ACC3.
204	4	CCVERFRR	Pointer to the FRR routine to protect RISGNL routines.
208	4	CCVEMKV	Address of the master key validate routine.
212	4	CCVEENFS	ECB for *Issue ENF SIGNAL*.
216	4	CCVETWIN	Address of CSFATWIN
220	4	CCVETWOT	Address of CSFATWOT
224	4	CCVEVKID	Pointer to CSFAVKID
228	4	CCVEMKVB	Pointer to the current Master Key Verification Pattern (MKVP) block. See “Master Key Verification Pattern Block (MKVB)” on page 177 for a description of the MKVP block.
232	32	CCVEMKB1	First MKVP block.
264	32	CCVEMKB2	Second MKVP block.
296	32	CCVEMKB3	Third MKVP block.
328	4	CCVEINPP	Pointer to installation optional parameter.
332	4	CCVEINPL	Length of the installation optional parameter.
336	4	CCVETRCN	Number of trace entries.
340	4	CCVESMIL	SMIB for large blocks.
344	4	CCVEPMKV	Address of CSFMPMKV.
348	16		Reserved.
364	4	CCVEWKAR	Work area for services.
368	4	CCVETMST	CPOOL ID for ASSPC.
372	8	CCVESECC	Reserved for security exit.
380	4	CCVEENTK	ENTE for security keys exit.
384	4	CCVEENTS	ENTE for security service exit.
388	4	CCVEIOST	Address of I/O subtask TCB.
392	4	CCVEIOPB	Address of I/O subtask data.
396	16	CCVE_PKA_KMMK_HP	KMMK hash pattern.
412	16	CCVE_PKA_SMK_HP	SMK hash pattern.
428	4	CCVEIOST_PKDS	Address of PKDS I/O subtask.
432	4	CCVEIOPB_PKDS	Address of PKDS I/O st data.
436	4	CCVEPKDL	Pointer to PKDS interface.
440	44	CCVEPKDS	Data set name of the PKDS.
484	4	CCVECCPD	Address of CAJ data.
488	4	CCVECCPV	Address of private CAJ data.
492	4	CCVEGSVT	Address of generic service vector table.
496	4	CCVEGSFL	GSVT flags.

Table 36. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
500	54	CCVEWLDS	Data set name of Wait List data set.
554	2	*	Padding.
556	4	CCVEMUST	Address of UDX service table.
560	4	CCVEQSCC	Queue State Change Count.
564	4	CCVEPKCH	Size of PKDS cache in records.
568	24	*	Reserved.
592	4	*	Reserved for alignment.
596	4		Ensure CCVE ends on a doubleword boundary.

Master Key Verification Pattern Block (MKVB)

Table 37 describes the contents of the MKVB.

Table 37. Master Key Verification Pattern Block Format

Offset (Dec)	Number of Bytes	Description
0	4	Pointer to the next element or zero.
4	4	Pointer to the next element — this field for use by CSFMMKV.
8	4	Reserved.
12	1	Master key version number for this verification pattern.
13	1	Flag. **Bit** **Meaning When Set On** **0** This element is on the active queue.
14	2	Reserved.
16	8	Master Key Verification Pattern.
24	8	Master Key Authentication Pattern.

Generic Service Table (CSFMGST)

Table 38 describes the format of the generic service table, a control block that is used to control the call of installation-defined services.

Table 38. Generic Service Table Block Format

Offset (Dec)	Number of Bytes	Description
0	4	EBCIDIC ID.
4	2	Version number.
6	2	Length of the MGST.
8	4	Number of entries in the array.
12	4	Subpool this table is in.
16	4	Reserved.
20	4	Reserved.
24	4	Reserved.
28	4	Reserved.

Table 38. Generic Service Table Block Format (continued)

Offset (Dec)	Number of Bytes	Description
Variable Section of the MGST		
32	8	IBM-assigned name.
40	8	Installation-assigned name.
48	4	Flags. **Bit** — **Meaning When Set On** **0** — Service has been requested by the installation. **1** — Service has been loaded. **2** — Service is active. **3** — Service is required.
52	4	Address of the service.
56	4	Installation-assigned service number.
60	4	Reserved.

RMF Measurements Table

Table 39 describes the contents of the performance measurements for RMF. The count fields are double-word length.

Table 39. RMF Measurements Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	4	DACC_ID	The DACC ID.
4	4	DACC_VER	The version.
8	4	DACC_LEN	The control block length.
12	4	DACC_ENT_CNT	Number of entries.
16	4	DACC_ENT_LEN	Length of each entry.
20	8	DACC_ENT_ID	Identifier of count array - character 'ENCSDES'. The Encipher service will collect data as follows: • Collection for single DES is done separately. The number of service calls, number of bytes of data enciphered, and the number of hardware instructions used to encipher the data will be collected.
28	8	DACC_ENT_SVC_CNT	Count of ENCSDES service calls.
36	8	DACC_ENT_BYT_CNT	Count of ENCSDES bytes processed.
44	8	DACC_ENT_INT_CNT	Count of ENCSDES instructions.
52	8	DACC_ENT_ID	Identifier of count array - character 'ENCTDES'. The Encipher service will collect data as follows: • Double and triple DES will be counted together. The number of service calls, number of bytes of data enciphered, and the number of hardware instructions used to encipher the data will be collected.
60	8	DACC_ENT_SVC_CNT	Count of ENCTDES service calls.
68	8	DACC_ENT_BYT_CNT	Count of ENCTDES bytes processed.
76	8	DACC_ENT_INT_CNT	Count of ENCTDES instructions.

Table 39. RMF Measurements Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
84	8	DACC_ENT_ID	Identifier of count array - character 'DECSDES'. The Decipher service will collect data as follows: • Collection for single DES is done separately. The number of service calls, number of bytes of data deciphered, and the number of hardware instructions used to decipher the data will be collected.
92	8	DACC_ENT_SVC_CNT	Count of DECSDES service calls.
100	8	DACC_ENT_BYT_CNT	Count of DECSDES bytes processed.
108	8	DACC_ENT_INT_CNT	Count of DECSDES instructions.
116	8	DACC_ENT_ID	Identifier of count array - character 'DECTDES'. The Decipher service will collect data as follows: • Double and triple DES will be counted together. The number of service calls, number of bytes of data deciphered, and the number of hardware instructions used to decipher the data will be collected.
124	8	DACC_ENT_SVC_CNT	Count of DECTDES service calls.
132	8	DACC_ENT_BYT_CNT	Count of DECTDES bytes processed.
140	8	DACC_ENT_INT_CNT	Count of DECTDES instructions.
148	8	DACC_ENT_ID	Identifier of count array - character 'MACGEN'. The MAC Generate service will collect data as follows: • Single and various double key MAC will be gathered together. The number of service calls, number of bytes of data MAC'd, and the number of instructions will be collected.
156	8	DACC_ENT_SVC_CNT	Count of MACGEN service calls.
164	8	DACC_ENT_BYT_CNT	Count of MACGEN bytes processed.
172	8	DACC_ENT_INT_CNT	Count of MACGEN instructions.
180	8	DACC_ENT_ID	Identifier of count array - character 'MACVER'. The MAC Verify service will collect data as follows: • Single and various double key MAC will be gathered together. The number of service calls, number of bytes of data MAC'd, and the number of instructions will be collected.
188	8	DACC_ENT_SVC_CNT	Count of MACVER service calls.
196	8	DACC_ENT_BYT_CNT	Count of MACVER bytes processed.
204	8	DACC_ENT_INT_CNT	Count of MACVER instructions.
212	8	DACC_ENT_ID	Identifier of count array - character 'OWH'. The One Way Hash service will collect data as follows: • For SHA-1, the number of service calls, number of bytes of bytes of data hashed, and the number of instructions will be collected.
220	8	DACC_ENT_SVC_CNT	Count of OWH service calls.
228	8	DACC_ENT_BYT_CNT	Count of OWH bytes processed.
236	8	DACC_ENT_INT_CNT	Count of OWH instructions.

Table 39. RMF Measurements Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
244	8	DACC_ENT_ID	Identifier of count array - character 'PTR'. The PIN Translate service will collect data as follows: • Collect the number of service calls only.
252	8	DACC_ENT_SVC_CNT	Count of PTR service calls.
260	16		Reserved.
276	8	DACC_ENT_ID	Identifier of count array - character 'PVR'. The PIN Verify service will collect data as follows: • Collect the number of service calls only.
284	8	DACC_ENT_SVC_CNT	Count of PVR service calls.
292	16		Reserved.

Appendix B. Installing the CICS-ICSF Attachment Facility

The purpose of the CICS-ICSF Attachment Facility is to enhance the performance of CICS transactions in the same region as a transaction using long-running ICSF services such as the PKA services and CKDS or PKDS update services. Without the CICS-ICSF Attachment Facility, the application that requests a long-running ICSF service is placed into an OS WAIT. This affects any other transactions that run in the same region. The CICS-ICSF Attachment Facility consists, in part, of a CICS Task-Related User Exit (TRUE). The TRUE attaches a task control block (TCB) which does the actual call to the ICSF service. This allows the CICS application that requests the long-running service to be placed into a CICS WAIT, rather than an OS WAIT, for the duration of the operation.

Before you can use the CICS-ICSF Attachment Facility, the ICSF system programmer, or the CICS administrator needs to install it. This involves the following steps:

- Relinking the ICSF enabling routine, CSFATREN, and the ICSF TRUE, CSFATRUE, if ICSF was previously installed in an environment without the CICS-ICSF Attachment Facility
- Installing the proper load libraries in the PROC used to start CICS
- Updating the CICS System Definitions (CSD) data set to define the programs to CICS
- Enabling these programs

For information about CICS TRUE programs, refer to *CICS Customization Guide*, SC33-1683.

1. If ICSF was previously installed in an environment without the CICS-ICSF Attachment Facility (i.e., without being linked with the CICS SDFHLOAD data set), the ICSF system programmer will need to relink the ICSF TRUE, CSFATRUE, and the ICSF enabling routine, CSFATREN. This would be the case if, for example, (a) the DDDEF entries for ICSF do not have the SDFHLOAD DDDEF pointing to the CICS SDFHLOAD data set but instead have it pointing to an empty data set, or (b) z/OS (and hence ICSF) was installed using a ServerPac.

   To relink the ICSF modules, first manually update the ICSF DDDEF for SDFHLOAD to point to the CICS SDFHLOAD data set. (Refer to ICSF sample CSFDDDEF shipped in SAMPLIB.) Then submit a job to relink the ICSF modules. The following is an example of job control language for the relink.

```
//STEP01        EXEC PGM=IEWL,
//   PARM='LIST,XREF,LET,DCBS,AMODE(31),RMODE(24)'
//SYSLMOD   DD DISP=SHR,DSN=yyy.SCSFMOD0   (the ICSF load library)
//SYSLIB    DD DISP=SHR,DSN=xxxxxx.SDFHLOAD
//SDFHLOAD  DD DISP=SHR,DSN=xxxxxx.SDFHLOAD
//SCSFMOD0  DD DISP=SHR,DSN=yyy.SCSFMOD0   (the ICSF load library)
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSPRINT  DD SYSOUT=*
//SYSLIN    DD *
       INCLUDE SDFHLOAD(DFHEAI)
       REPLACE CSFDHEAI(DFHEAI),CSF0EAI
       INCLUDE SCSFMOD0(CSFATREN)
         ENTRY DFHEAI
     NAME CSFATREN(R)
       INCLUDE SDFHLOAD(DFHEAI)
       REPLACE CSFDHEAI(DFHEAI),CSF0EAI
```

```
       INCLUDE SCSFMOD0(CSFATRUE)
         ENTRY DFHEAI
     NAME CSFATRUE(R)
/*
```

2. Include the ICSF load module data set in the CICS startup job control language as shown in the following example.

```
//DFHRPL    DD DISP=SHR,DSN=xxxxx.SDFHLOAD
//          DD DISP=SHR,DSN=yyy.SCSFMOD0 (The ICSF load library)
//          DD ...
...
//SYSIN     DD DISP=SHR,DSN=xxxxx.SYSIN(DFH$SIPx)
...
```

   In the above sample code, DFH$SIPx includes the entry:

```
PLTPI=yy,
```

3. Customize the Program Load Table (PLT), to include the ICSF enabling routine CSFATREN in second stage initialization.

   The following is an example input deck for compiling a PLT for automatic enablement of the CICS-ICSF link. This is ASM code. Assemble it with the CICS macro library, but **without** the CICS translator.

```
//SYSIN DD *
*
* List of programs to be executed sequentially during system
* initialization.  Required system initialization parm: PLTPI=yy
* DFHPLTCS should be defined in the CSD by CEDA or DFHCSDUP job
*
DFHPLT TYPE=INITIAL,SUFFIX=yy
*
* -------- Second stage of initialization -----------------
*
DFHPLT TYPE=ENTRY,PROGRAM=CSFATREN  (Run enable of CSFATRUE)
*
* ---------- Delimiter between Stages 2 and 3 ------------
*
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
* --------- Third stage of initialization -----------------
* (none)
*
DFHPLT TYPE=FINAL
END
/*
```

   The above code is an example only. Your CICS administrator can use it as a guide in customizing the PLT. For more information about coding the PLT, refer to *CICS Resource Definition Guide*.

4. Link edit the PLT with the following controls:

```
INCLUDE OBJLIB(DFHPLTyy)
NAME    DFHPLTyy(R)
```

5. The CICS administrator should customize the system CSD to include the following:
   - CSFATRUE
   - CSFATREN
   - A PLT to indicate that initialization is to call CSFATREN to enable the ICSF TRUE, CSFATRUE

The following is an example of the job control language and input. In this example, xxxxx represents the local CICS prefix, and zzzzzzzz represents the PLT entry that was compiled above.

```
//UPDATE  JOB  ...
//*- - - - - - - - - - - - - - - - - - - - -
//DEFINES EXEC PGM=DFHCSDUP,REGION=2M
//STEPLIB   DD DISP=SHR,DSN=xxxxxx.SDFHLOAD
//          DD DISP=SHR,DSN=zzzzzzzz
//DFHCSD    DD DISP=SHR,DSN=xxxxxx.DFHCSD
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
*
DEFINE PROGRAM(CSFATREN) GROUP(ICSF)
                         DESCRIPTION(TRUE enablement routine)
                         LANGUAGE(ASSEMBLER)
*
DEFINE PROGRAM(CSFATRUE) GROUP(ICSF)
                         DESCRIPTION(ICSF interface TRUE)
                         LANGUAGE(ASSEMBLER)
*
DEFINE PROGRAM(DFHPLTyy) GROUP(ICSF)
                         DESCRIPTION(PLT Program Init for CSFATRUE)
                         LANGUAGE(ASSEMBLER)
```

The PLT in the example runs the program CSFATREN during CICS initialization. CSFATREN automatically enables the ICSF TRUE, CSFATRUE. If CICS is already started, use a CICS Command Level Interpreter Transaction (CECI) to enable CSFATRUE. To do this, go into CECI and issue the following statement:

```
ENABLE PROGRAM('CSFATRUE') TALENGTH(140) LINKEDITMODE START
```

You can also do this in a single step with the following statement:

```
CECI ENABLE PROGRAM('CSFATRUE') TALENGTH(140) LINKEDITMODE START
```

6. Relink any existing CICS transactions that call any of the following ICSF services. (Sample JCL for linking the ICSF callable services into an application program can be found in ″Linking a Program with the ICSF Callable Services″ in the *z/OS ICSF Application Programmer's Guide*, SA22-7522.)
   - CSFPCI — PCI Interface
   - CSNBCPA — Clear PIN Generate Alternate
   - CSNBCPE — Clear PIN Encrypt
   - CSNBCSG — VISA CVV Service Generate
   - CSNBCSV — VISA CVV Service Verify
   - CSNBCVE — Cryptographic Variable Encipher
   - CSNBCVT — Control Vector Translate
   - CSNBDKG — Diversified Key Generate
   - CSNBDKM — Data Key Import
   - CSNBDKX — Data Key Export
   - CSNBEPG — Encrypted PIN Generate
   - CSNBKEX — Key Export
   - CSNBKGN — Key Generate
   - CSNBKIM — Key Import
   - CSNBKPI — Key Part Import
   - CSNBKRC — CKDS Key Record Create
   - CSNBKRD — CKDS Key Record Delete
   - CSNBKRW — CKDS Key Record Write
   - CSNBKTR — Key Translate
   - CSNBKYT — Key Test
   - CSNBPEX — Prohibit Export
   - CSNBPGN — Clear PIN Generate

- CSNBPTR — Encrypted PIN Translate
- CSNBPVR — Encrypted PIN Verify
- CSNBSKI — Secure Key Import
- CSNBSKM — Multiple Secure Key Import
- CSNBSKY — Secure Messaging for Keys
- CSNBSPN — Secure Messaging for PINs
- CSNDDSG — Digital Signature Generate
- CSNDDSV — Digital Signature Verify
- CSNDKRC — PKDS Record Create
- CSNDKRD — PKDS Record Delete
- CSNDKRR — PKDS Record Read
- CSNDKRW — PKDS Record Write
- CSNDKTC — PKA Key Token Change
- CSNDPKB — PKA Key Token Build
- CSNDPKD — PKA Decrypt
- CSNDPKE — PKA Encrypt
- CSNDPKG — PKA Key Generate
- CSNDPKI — PKA Key Import
- CSNDPKX — PKA Public Key Extract
- CSNDRKD — Retained Key Delete
- CSNDRKL — Retained Key List
- CSNDSBC — SET Block Compose
- CSNDSBD — SET Block Decompose
- CSNDSYG — Symmetric Key Generate
- CSNDSYI — Symmetric Key Import
- CSNDSYX — Symmetric Key Export

Beginning in OS/390 V2 R10 ICSF, the CICS Wait List can be implemented by means of a customer modifiable data set, pointed to by the Installation Options Data Set (WAITLIST parameter). The default WAITLIST includes all services noted above. If the new option is not specified, the default CICS Wait List will be utilized by ICSF when a CICS application invokes an ICSF callable service. If WAITLIST is specified, the data set specified by this parameter will be used to determine the names of the services to be placed on the CICS Wait List. A sample data set is provided by ICSF via member CSFWTL00 of SYS1.SAMPLIB. The sample data set contains the same entries as the default ICSF CICS Wait List -- for example, the data set contains the names of all ICSF callable services which, by default, will be driven through the CICS TRUE. The following is a sample installation data set that contains the WAITLIST parameter.

```
CKDSN(CSF.SCSFCKDS)
PKDSN(CSF.SCSFPKDS)
PKDSCACHE(64)
COMPAT(NO)
SSM(YES)
KEYAUTH(NO)
CHECKAUTH(NO)
WAITLIST(HARVILL.CSFWTL00.SAM)
TRACEENTRY(599)
COMPENC(DES)
```

The WAITLIST option should be added to the Installation Options data set under the following conditions.

- Non-CICS customers will not specify a WAITLIST keyword.
- CICS customers who want to use the default CICS Wait List shipped with OS/390 V2 R10 ICSF will not specify a WAITLIST keyword. You must ensure, however, that any existing CICS applications which invoke any of the ICSF services in the Wait List are re-linked to pick up the new version of the stub.

- CICS customers who do not want to make use of CICS TRUE must either not enable the TRUE or specify a WAITLIST keyword and point to an empty wait list data set or you can specify WAITLIST(DUMMY) in the Installation Options data set.
- CICS customers who wish to modify the ICSF default CICS Wait List should modify the sample Wait List data set supplied in member CSFWTL00 of SYS1.SAMPLIB. The WAITLIST keyword in the Installation Options Data Set should be set to point to this data set. Any existing CICS applications which invoke any of the ICSF services in the Wait List should be re-linked to pick up the new version of the stub.

If you already have the CICS-ICSF Attachment facility installed, there are a number of callable services in OS/390 V2 R10 ICSF and higher which may potentially be routed to the PCI Cryptographic Coprocessor for processing. The services which may be routed to the PCI Cryptographic Coprocessor must be added to the CICS Wait List and relinked. You can use the default CICS Wait List that is shipped with ICSF which includes all services which have an asynchronous interface to ICSF or you can use a sample Wait List data set that is also shipped with ICSF. The sample CICS Wait List data set is contained in member CSFWTL00 of SYS1.SAMPLIB. The sample data set contains the same entries as the default ICSF CICS Wait List. You can modify the sample data set to add and/or delete items from the Wait List. Here are some examples of why you might want to modify the sample data set:

- If you do not have a PCI Cryptographic Coprocessor installed, you can delete all of the services identified with an ″*″ that are in the sample wait list.
- If you have a PCI Cryptographic Coprocessor installed, you can examine the services your applications invoke in a CICS environment and determine, based upon the routing information provided for each service in *z/OS ICSF Application Programmer's Guide*, SA22-7522, that the service will never be routed to a PCI Cryptographic Coprocessor. In this case (except for the CKDS/PKDS access services) the service can be deleted from the list.
- If you have an application which invokes a UDX while running under CICS, then the name of the UDX generic service should be added to the CICS Wait List.

If you use a CICS Wait List data set, you need to identify the data set to ICSF through the WAITLIST(data_set_name) option in the ICSF Installation Options data set. The data set can be a member of a PARMLIB, a member of a partitioned data set, or a sequential data set. The data set should be allocated on a permanently resident volume and should adhere to the following:

- The format of each record in the data set must be fixed length or fixed block length.
- A physical line in the data set must be a LRECL of 80 characters long. The system ignores any characters in positions 73 to 80 of the line.
- You can delimit comments by ″/*″ and ″*/″ and include them anywhere in the text. A comment cannot span physical records.
- Only one service may be specified on a logical line.

Note: You can use the WAITLIST(DUMMY) parameter to specify a null CICS Wait List data set, or you can disable the CICS TRUE if you do not want to utilize the CICS TRUE. See “Changing Parameters in the Installation Options Data Set” on page 22 for additional information.

Appendix C. Helpful Hints for ICSF First Time Startup

The purpose of this section is to provide some helpful hints and resolutions for the problems that you may encounter when starting ICSF for the first time.

See Appendix E, "Running HCR7708 on a IBM @server zSeries 990", on page 195 if you're running in this environment.

Checklist for First-Time Startup of ICSF

The following is a checklist for the first-time startup of ICSF.

Step 1. Hardware Setup

Process Crypto Enablement Diskette Load

PCICC FCV Load (if applicable)

Power-on Reset

Responsible CE or Client Operator Representative

Where Support Element

Verify Via Cryptographic Coprocessor Configuration Task

- Status for CP0 and/or CP1 is ″Initialized″

Via PCI Cryptographic Coprocessor Configuration Task

- Status for Pxx is ″Configured″

References *Support Element Operations Guide*

Completed

Step 2. LPAR Activation Profiles

Process Crypto Page Setup

PCICC Page Setup

Processor Page Setup

Responsible CE or Client Operator Representative

Where Support Element

Verify On Crypto Page of the Activation Profile

- Enable Public Key Algorithm
- Enable Cryptographic Functions
- Enable PKSC and ICSF
- Enable Cryptographic Facility (ICRF) Key Entry
- Enable Special Secure Mode

If using TKE, also

- Enable Modify Authority (Only (1) LPAR - TKE Host)
- Enable Query Signature Controls (TKE Host)
- Enable Query Transport Controls (TKE Host)

- For the TKE Host, the Control Domain must include ALL the domains that will be controlled by the TKE Host

On the PCICC Page of the Activation Profile

- PCI Cryptographics Coprocessor Candidate List includes all PCICC's that CAN be online
- PCI Cryptographics Coprocessor Online List includes all PCICC's that WILL be online when activation is complete (Selections in the Online List MUST be selected in the Candidate List)

On the Processor Page Setup

- Cryptographic Coprocessor(s) are enabled for that LPAR

References *Support Element Operations Guide*

z/OS ICSF TKE Workstation User's Guide 2000, SA22-7524 (LPAR Considerations)

S/390 PR/SM Planning Guide

Completed

Step 3. ICSF Setup

Process Install and Customize ICSF

Responsible System Programmer and ICSF Administrator

Where TSO and ISPF Panels

Verify Customize SYS1.PARMLIB

- Add CSF.SCSFMOD0 to the LNKLST concatenation
- Update PROGxx to APF authorize CSF.SCSFMOD0
- Update IKJTSOxx for ICSF by adding CSFDAUTH to the AUTHPGM and AUTHTSF parameter lists

CKDS and PKDS created

ICSF Startup Procedure created

Installation Options Dataset created

- Beginning in z/OS V1 R2, the DOMAIN parameter in the installation options data set is optional. It is required if more than one domain is specified as the usage domain on the PR/SM panels or if running in native mode.
- CKDS and PKDS names specified
- COMPAT(NO) and SSM(YES)

Access provided to the ICSF panels

References Chapter 2, "Installation, Initialization, and Customization", on page 9

Completed

Step 4. TKE Setup

If you are not using TKE, proceed to the next step.

Process Initialize the TKE Workstation

Configure TCP/IP on the Host and the TKE Workstation

Setup the TKE Host Transaction Program

- Create JCL to start the TKE Host Transaction Program
- RACF Security Setup
- Start the TKE Host Transaction Program

Responsible Network Programmer, System Programmer and TKE Administrator

Where ISPF Panels, TKE Workstation

Verify CSFTTKE is authorized in the AUTHCMD list of IKJTSOxx in SYS1.PARMLIB

TKE Host Transaction Program (CSFTTCP) is defined in the RACF STARTED class (If your installation has a Generic Userid associated to all started procedures, this is not necessary)

CSFTTKE profile is defined in the RACF FACILITY and RACF APPL classes

References *z/OS ICSF TKE Workstation User's Guide 2000*, SA22-7524 (See Topics: TKE Workstation Setup and Customization and TKE TCP/IP and Host Considerations)

Completed

Step 5. ICSF Startup

Process Start ICSF

Responsible Client Operator Representative or System Programmer

Where Operator Console

References Chapter 2, "Installation, Initialization, and Customization", on page 9

Completed

Step 6. Loading Master Keys and Initializing the CKDS through ICSF Panels

If you are using TKE, proceed to the next step.

Process Passphrase Initialization to load and SET master keys and initialize CKDS

- Create NOCV, ANSI, and ESYS keys as applicable for your installation

- or -

Clear Master Key Entry

- Load DES New Master Key
- Load PKA Signature Master Key (SMK)
- Load PKA Key Management Master Key (KMMK)
- Load New Symmetric Master Key (if applicable)
- Load New Asymmetric Master Key (if applicable)

Note: Using the Coprocessor Management panel, the master keys can be loaded into all the coprocessors (Cryptographic Coprocessor Feature and PCI

Cryptographic Coprocessor) at the same time. It is recommended that the SMK and KMMK keys be set to the same value.

- Initialize CKDS and SET the DES New Master Key
- Create NOCV, ANSI, and ESYS keys as applicable for your installation
- Enable PKA Services
- Enable PKDS Read Access
- Enable PKDS Write, Create, and Delete Access

Responsible ICSF Administrator and Key Officers

Where ICSF Panels

Verify In System Log:

- IEE504I CRYPTO(0),ONLINE
- IEE504I CRYPTO(1),ONLINE (if applicable)
- CSFM116I BOTH MASTER KEYS CORRECT ON PCI CRYPTOGRAPHIC COPROCESSOR Pnn, SERIAL NUMBER nn-nnnn (if applicable)
- CSFM400I CRYPTOGRAPHY SERVICES ARE NOW AVAILABLE

References *z/OS ICSF Administrator's Guide*, SA22-7521 (See Topics: Using the Pass Phrase Initialization Utility, Managing Master Keys on the S/390 Enterprise Servers, and Managing Master Keys on the S/390 Enterprise Server)

Completed

Step 7. Customizing TKE and Loading Master Keys

If you are not using TKE, proceed to the next step.

Process TKE Administrator's and Key Officers

- Define Host ID's
- Define CCF Authorities
- Define Access Controls (Signature Requirements for CCF)
- Define Roles (if applicable)
- Define PCI Cryptographic Coprocessor Authorities (if applicable)
- Load DES New Master Key
- Load PKA Signature Master Key (SMK)
- Load PKA Key Management Master Key (KMMK)
- Load New Symmetric Master Key (if applicable)
- Load and SET New Asymmetric Master Key (if applicable)

Note: If you have more than one crypto module or PCI Cryptographic Coprocessor, repeat the process for each, unless Groups have been defined. It is recommended that the SMK and KMMK keys be set to the same value.

Responsible ICSF Administrator

- Initialize CKDS and SET the DES New Master Key
- Create NOCV, ANSI, and ESYS keys as applicable for your installation

- Enable PKA Services
- Enable PKDS Read Access
- Enable PKDS Write, Create, and Delete Access

Where TKE Workstation and ICSF Panels

Verify In System Log:

- IEE504I CRYPTO(0),ONLINE
- IEE504I CRYPTO(1),ONLINE (if applicable)
- CSFM116I BOTH MASTER KEYS CORRECT ON PCI CRYPTOGRAPHIC COPROCESSOR Pnn, SERIAL NUMBER nn-nnnn (if applicable)
- CSFM400I CRYPTOGRAPHY SERVICES ARE NOW AVAILABLE

References

z/OS ICSF Administrator's Guide, SA22-7521 (See Topics: Managing Master Keys on the S/390 Enterprise Servers)

Completed

Step 8. CICS-ICSF Attachment Facility Setup

If you are not using CICS-ICSF, proceed to the next section.

Process Follow the instructions in Appendix B, "Installing the CICS-ICSF Attachment Facility", on page 181.

Responsible System Programmer

Where SMP/E Panels and Sample Jobs

References Appendix B, "Installing the CICS-ICSF Attachment Facility", on page 181

Completed

Normal ICSF Messages at First Time Startup

It is normal to see the following error and informational messages during first time startup of ICSF.

```
CSFM100E CRYPTOGRAPHIC KEY DATA SET, ECHAN.DOMAIN7.CKDS IS NOT
INITIALIZED.
CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR 0, CPU 0 IS NOT VALID.
CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR 1, CPU 4 IS NOT VALID.
CSFM106A CRYPTOGRAPHY - PKA MASTER KEYS ARE NOT VALID.
CSFM411I PCI CRYPTOGRAPHIC ACCELERATOR Axx IS ACTIVE.
CSFM001I ICSF INITIALIZATION COMPLETE
```

Commonly Encountered ICSF First Time Setup/initialization Messages

The following ICSF messages are commonly encountered during initialization and first time startup of ICSF.

- **CSFM105E CRYPTOGRAPHY - DOMAIN '***domain***' IS NOT ACCESSIBLE** - A domain mismatch exists between the domain you have selected in your LPAR activation profile and the domain option specified in your ICSF options data set. You must decide which domain is the one you want and correct it in the appropriate location.

- **CSFM107E CRYPTOGRAPHY - CRYPTO MODULES CONFIGURED DIFFERENTLY** - The CCC values on both of your crypto coprocessors must be the same. One of the cryptos may not have been loaded with an enablement diskette yet, or selected for next activation with the force zeroize option. Ensure that both crypto coprocessors are loaded with the same configuration. An IPL will be required.
- **CSFM120E PUBLIC KEY SECURE CABLE (PKSC) FACILITY IS NOT ENABLED** - The Enable cryptographic functions option and/or the Enable public key secure cable (PKSC) and integrated cryptographic service facility (ICSF) option is not enabled in the LPAR activation profile. Check the appropriate boxes to enable the options.
- **CSFM410E ERROR IN OPTIONS DATA SET** - ICSF could not interpret the options data set.

Before trying to start ICSF, ensure that the crypto coprocessors have been initialized with the enablement diskette. If the coprocessors have been loaded, a configuration should be available to select for next activation from the Cryptographic Coprocessors Configuration panels. If the crypto coprocessors have not been loaded with the enablement diskette and ICSF is started, message CSFM107E will be issued. This message will only be issued if you have 2 Cryptographic Coprocessor Features and they do not contain the same CCC. If the CCCs have not been initialized (are all zeroes) you will receive an X'18F' reason code 4a abend.

Starting with OS/390 V2 R9 ICSF, a pre-allocated PKDS is required. The PKDS data set name must be specified in the options data set with the PKDSN option. If a PKDS is not specified, you will receive the following messages:

```
CSFM408A NO PKDS NAME WAS SPECIFIED IN THE OPTIONS DATA SET.
CSFM401I CRYPTOGRAPHY - SERVICES NO LONGER AVAILABLE.
```

Appendix D. Using AMS REPRO Encryption

This appendix provides information on using IDCAMS REPRO ENCIPHER and DECIPHER options with ICSF.

Restriction: AMS REPRO Encryption is not supported running FMID HCR7708 on an IBM @server zSeries 990.

Steps for setting up ICSF

Perform the following tasks to use the ENCIPHER and DECIPHER parameters with ICSF:

1. Define the key value that is used to encrypt and decrypt the data key. To define the key value, use one of the following ICSF key administrative options:
   - Trusted Key Entry (TKE) workstation. For information about how to define the key value using the TKE workstation, see *z/OS ICSF TKE Workstation User's Guide 2000*.
   - Key generator utility program (KGUP). Use the KGUP panel ″ICSF - Create ADD, UPDATE, or DELETE Key Statement″ to define the key value. For more information about how to use KGUP panels, see *z/OS ICSF Administrator's Guide*.

     Restrictions:

     – The length of the data encryption key is limited to 8 bytes, or 56-bit DES. Triple DES support is not available.

     – Key labels are limited to 8 characters because of the fixed size of REPRO storage areas.

     – The REPRO command's encryption algorithm variables are not documented, so you cannot use them to write decryption applications on another system. Therefore, cross-platform exchange is not possible.
2. Refresh ICSF's cryptographic key data set (CKDS) so that the key value can be used by REPRO.
3. Ensure that ICSF can support PCF macro calls by specifying COMPAT(YES) in the ICSF installation options. For more information about how to specify ICSF installation options, see Chapter 2, “Installation, Initialization, and Customization” , on page 9.

   If you had to change the ICSF installation options, you must restart ICSF.
4. Run the REPRO ENCIPHER or DECIPHER job.

Restrictions: The REPRO command's encryption algorithm variables are not documented, so you cannot use them to write decryption applications on another system. Therefore, cross-platform exchange is not possible.

Recommendation:

1. Do not specify the REPRO parameter PRIVATEKEY, because it exposes the clear data key value. Instead, specify either EXTERNALKEY or INTERNALKEY, and STOREDATAKEY.

Appendix E. Running HCR7708 on a IBM @server zSeries 990

Support for the IBM @server zSeries 990 has been added. Crypto assist instructions and the optional PCI Cryptographic Accelerator are available on this server. This server does not support the Cryptographic Coprocessor Feature or the PCI Cryptographic Coprocessor.

During the initialization of HCR7708, the startup task determines the type of hardware environment. If the Cryptographic Coprocessor Feature is available, processing continues the same as in the previous release of ICSF and no change is needed for existing Cryptographic Coprocessor Feature configurations. If crypto assist instructions or the optional PCI Cryptographic Accelerator is available, without the Cryptographic Coprocessor Feature for secure cryptography, the machine type will be recognized as the IBM @server zSeries 990.

Running HCR7708 on the IBM @server zSeries 990 differs from running it on the S/390 G5 Enterprise Server, S/390 G6 Enterprise Server, IBM @server zSeries 800 and IBM @server zSeries 900. This section describes the processing differences in the IBM @server zSeries 990 environment. This section does not apply if you are running HCR7708 on a S/390 G5 Enterprise Server, S/390 G6 Enterprise Server IBM @server zSeries 800 or IBM @server zSeries 900.

Operating System Requirements

HCR7708 can be installed on z/OS V1 R3 or z/OS V1 R4 if you are installing on the IBM @server zSeries 990. System SSL and Communication Server applications are the only applications that you will be able to exploit. Exploitation of System SSL on z/OS V1 R3 requires the System SSL web deliverable available with the ICSF web deliverable, HCR7708.

Applications and programs

Applications requiring secure cryptography using encrypted keys will not be able to execute on the IBM @server zSeries 990. All cryptographic keys must be clear keys.

The following application and programs are not supported on the IBM @server zSeries 990:

- Access Method Services Cryptographic option
- CICS attachment facility
- CKDS Conversion program
- CSFEUTIL program for CKDS reencipher, refresh, change master key, and passphrase initialization functions
- Distributed Key Management System (DKMS)
- Key Generation Utility Program (KGUP)
- PCF applications
- PKDS activate program (CSFPUTIL)
- PKDS cache refresh program (CSFPUTIL)
- PKDS reencipher program (CSFPUTIL)
- UDX (User Defined Extension) support
- VTAM Session Level Encryption

- 4753-HSP applications

Callable services

The following services are available when running HCR7708 on a IBM @server zSeries 990:

- Character/Nibble Conversion (CSNBXBC and CSNBXCB)
- Code Conversion (CSNBXEA and CSNBXAE)
- Control Vector Generate (CSNBCVG)
- Decode (CSNBDCO) - This service requires enablement of CP Assist for Cryptographic Functions.
- Encode (CSNBECO) - This service requires enablement of CP Assist for Cryptographic Functions
- MDC Generate (CSNBMDG and CSNBMDG1) - This service requires enablement of CP Assist for Cryptographic Functions
- One–Way Hash Generate (CSNBOWH and CSNBOWH1)
- PKA Decrypt (CSNDPKD) - This service requires a PCI Cryptographic Accelerator.
- PKA Encrypt (CSNDPKE) ZERO-PAD formatting only - This service requires a PCI Cryptographic Accelerator.
- PKA Key Token Build (CSNDPKB)
- PKA Public Key Extract (CSNDPKX)
- Symmetric Key Decipher (CSNBSYD and CSNBSYD1) - This service requires enablement of CP Assist for Cryptographic Functions.
- Symmetric Key Encipher (CSNBSYE and CSNBSYE1) - This service requires enablement of CP Assist for Cryptographic Functions.
- X9.9 Data Editing (CSNB9ED)

Installation defined callable services are supported only if you're using clear keys and using one of the above supported callable services.

Callable services that require secure cryptography (the Cryptographic Coprocessor Feature and PCI Cryptographic Coprocessor) to execute, will fail with return code 12 and reason code 8 (Service or algorithm is not available on current hardware).

CKDS and PKDS

The CKDS (Cryptographic Key Data Set) and PKDS (Public Key Data Set) do not have to be allocated on a IBM @server zSeries 990.

CKDS and PKDS labelnames are not supported.

Exits

The following exit types are supported on a IBM @server zSeries 990:

- Mainline exits
- Exits for callable services

  Exits are only supported on available callable services. See "Callable services" for a list of available callable services.
- Security exits

ICSF Setup and Initialization

The following steps do not have to be performed when installing and initializing HCR7708 on a IBM @server zSeries 990:

- ICSF Setup
  - Create the Cryptographic Key Data Set (CKDS)
  - Create the Public Key Data Set (PKDS)
  - MK initialization for SMP/E
- TKE Setup
- Loading Master Keys and Initializing the CKDS through ICSF panels
- Customizing TKE and loading master keys
- CICS-ICSF attachment facility setup

It is normal to see the following messages during the startup of ICSF on a IBM @server zSeries 990:

- Starting ICSF on a IBM @server zSeries 990 without a PCI Cryptographic Accelerator.

  ```
  CSFM001I ICSF INITIALIZATION COMPLETE
  ```

- Starting ICSF on a IBM @server zSeries 990 with a PCI Cryptographic Accelerator. You 'll receive message CSFM411I for each PCI Cryptographic Accelerator you have online.

  ```
  CSFM411I PCI CRYPTOGRAPHIC ACCELERATOR Ann IS ACTIVE
  CSFM001I ICSF INITIALIZATION COMPLETE
  ```

Installation options data set

The following installation options data set keywords are supported on a IBM @server zSeries 990. All other keywords will be processed for syntax only and will otherwise be ignored.

- DOMAIN - On a IBM @server zSeries 990, if a PCI Cryptographic Accelerator is not available, the DOMAIN parameter is not required. If a PCI Cryptographic Accelerator is available, the DOMAIN parameter is required if more than one domain is specified as the usage domain on the PR/SM panels. If only one usage domain is assigned to the LPAR, the DOMAIN parameter is optional.
- CHECKAUTH
- TRACEENTRY
- USERPARM
- EXIT

  Exits are only supported on available callable services. See "Callable services" on page 196 for a list of available callable services.

Note: Optional keyword fields that are not specified will display as blanks on the ISPF panels.

Exploitation

If you are installing HCR7708 on a IBM @server zSeries 990 server, System SSL and some Communication Server applications are the only applications that you'll be able to exploit. Exploitation of System SSL on z/OS V1 R3 requires the System SSL web deliverable available with the ICSF web deliverable, HCR7708.

Note: There are no issues for installing HCR7708 on the following servers: S/390 G5 Enterprise Server, S/390 G6 Enterprise Server, IBM @server zSeries 800 and IBM @server zSeries 900.

Secure Sockets Layer (SSL)

System SSL applications are supported on the IBM @server zSeries 990. SSL defines methods for data encryption, server authentication, message integrity, and client authentication for a TCP/IP connection. Security is provided on the link and callable services that have been enhanced for DES, TDES and SHA-1 services. Exploitation of System SSL on z/OS V1 R3 requires the System SSL web deliverable available with the ICSF web deliverable, HCR7708.

TKE workstation

The Trusted Key Entry (TKE) workstation is not needed on the IBM @server zSeries 990. It is not needed as there are no hardware protected master keys.

TSO panels

Functions that are not available on a IBM @server zSeries 990 will appear disabled on the TSO panels.

Appendix F. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

This information was developed for products and services offered in the USA.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

Licensees 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
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This book primarily documents information that is NOT intended to be used as a Programming Interface of ICSF.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of ICSF. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

End of Programming Interface information

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
AT
CICS
CT
DFSMS/MVS
ES/3090
ES/9000
eServer
IBM
IBMLink
Multiprise
MVS
MVS/DFP
MVS/ESA
OS/390
Parallel Sysplex

Personal Security
Processor Resource/Systems Manager
PR/SM
RACF
Resource Link
RMF
S/390
S/390 Parallel Enterprise Server
System/370
System/390
VTAM
zSeries
z/OS
z/OS.e
3090

The following terms are trademarks or registered trademarks of other companies:

MasterCard	MasterCard International, Incorporated
SET	SET Secure Electronic Transaction, LLC
UNIX	The Open Group
VISA	VISA International Service Association

Other company, product, and service names may be trademarks or service marks of others.

Index

Numerics

A

C

D

E

F

G

I

K

L

M

N

O

P

R

S

T

U

V

W

Readers' Comments — We'd Like to Hear from You

z/OS
Integrated Cryptographic Service Facility
System Programmer's Guide

Publication No. SA22-7520-04

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

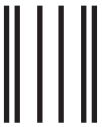
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SA22-7520-04

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Program Number: 5964-A01, 5655-G52

Printed in U.S.A.

SA22-7520-04