

z/OS
Cryptographic Services
Integrated Cryptographic Service Facility



System Programmer's Guide

z/OS
Cryptographic Services
Integrated Cryptographic Service Facility



System Programmer's Guide

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 239.

Thirteenth Edition, February 2008

This is a major revision of SA22-7520-11.

This edition applies to Version 1 Release 9 of z/OS (5694-A01) 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
MHVRCFS, Mail Station P181
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: mhvrdfs@us.ibm.com

World Wide Web: <http://www.ibm.com/systems/z/os/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.

© Copyright International Business Machines Corporation 1997, 2008. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	ix
Tables	xi
About this information	xiii
Who should use this information	xiii
How to use this information	xiii
Where to find more information	xv
Do You Have Problems, Comments, or Suggestions?	xvii
Summary of changes	xix
Chapter 1. Introduction to z/OS ICSF	1
Hardware Features	1
Cryptographic Hardware	1
Server Hardware	3
z/OS ICSF FMIDs	6
ICSF Features	7
The Cryptographic Key Data Set (CKDS)	8
The Public Key Data Set (PKDS)	9
The Token Data Set (TKDS)	9
Additional Background Information	10
Running PCF applications on z/OS ICSF	10
Running 4753-HSP applications on ICSF	10
Using RMF and SMF to monitor z/OS ICSF events	10
Controlling access to ICSF	11
Steps prior to starting installation	11
Chapter 2. Installation, Initialization, and Customization	13
Steps for installation and initialization	13
Steps to customize SYS1.PARMLIB	14
Steps to create the CKDS	15
Steps to create the PKDS	16
Steps to create the TKDS	19
Steps to create the Installation Options Data Set	21
Steps to create the ICSF Startup Procedure	23
Steps to provide access to the ICSF panels	24
Steps to start ICSF for the first time	26
MK Initialization for SMP/E - CCF Systems Only	30
Customizing ICSF after the first start	32
Changing parameters in the installation options data set	32
Improving CKDS performance	42
Dispatching priority of ICSF	42
Creating ICSF exits and generic services	42
Chapter 3. Migration	43
Terminology	43
Migrating from earlier software releases	44
Callable Services	44
ICSF Key Data Sets	48
Installation Options Data Set	49
Function Restrictions	50
CICS Attachment Facility	50

Special Secure Mode	51
Resource Manager Interface (RMF)	51
TKE Workstation	51
Migrating from the IBM @server zSeries 800 and the IBM @server zSeries 900	53
Callable Services	53
Functions Not Supported	53
Setup Considerations	53
Programming Considerations.	54
Migrating from 4753-HSP	55
Chapter 4. Operating ICSF	57
Starting and stopping ICSF	58
Modifying ICSF.	59
Using different configurations.	59
Configuring the IBM @server zSeries 990, IBM @server zSeries 890, IBM System z9 EC, IBM System z9 BC and System z10 Enterprise Class	59
Configuring the IBM @server zSeries 800 and the IBM @server zSeries 900	61
Adding and Removing Cryptographic Coprocessors	63
Adding Cryptographic Coprocessors	64
Steps for activating/deactivating cryptographic coprocessors	64
Steps to configure on/off cryptographic coprocessors	65
Steps for enabling/disabling cryptographic coprocessors (PCICC and PCIXCC/CEX2C)	65
Steps for enabling/disabling cryptographic coprocessors (CCF)	66
Performance considerations for using installation options	67
Dispatching priority of ICSF	68
VTAM session-level encryption	68
System SSL encryption	68
Access method services cryptographic option	69
Remote Key Loading.	69
Event Recording	70
System Management Facilities (SMF) Recording	70
Message Recording	77
Security Considerations.	77
Controlling the program environment	77
Controlling access to KGUP	78
Controlling access to the callable services	78
Controlling access to cryptographic keys	78
Scheduling changes for cryptographic keys	79
Controlling access to administrative panel functions	79
Controlling access to RACF SMF log records.	79
Debugging Aids	79
Component Trace	80
ICSF System SVC 143	82
Abnormal Endings.	83
IPCS Formatting Routine	83
Chapter 5. Installation Exits	87
Types of exits	87
Mainline exits	87
Exits for the services.	88
The PCF CKDS conversion program exit	88
The Single-record, Read-write exit.	88
The cryptographic key data set entry retrieval exit	88

Security exits	89
The KGUP exit	89
Entry and return specifications	89
Registers at entry	89
Registers at return	90
Exits environment	90
Mainline exits	91
service exits	91
CKDS entry retrieval exit	91
KGUP, Conversion Programs, and Single-record, Read-write exits	91
Security exits	91
Exit recovery.	91
Mainline installation exits	92
Purpose and use of the exits.	92
Environment of the exits	93
Installing the exits	93
Input.	94
Return Codes	99
Services installation exits	99
Purpose and use of the exits	100
Environment of the exits	100
Installing the exits	100
Input	104
Return Codes	109
Cryptographic key data set entry retrieval installation exit	109
Purpose and use of the exit.	110
Environment of the exit	110
Installing the exit	110
Input	111
Return codes	112
PCF conversion program installation exit	112
Purpose and use of the exit.	112
Environment of the exit	112
Installing the exit	113
Input	113
Return codes	114
Single-record, Read-write installation exit	115
Purpose and use of the exit.	115
Environment of the exit	116
Installing the exit	116
Input	116
Return codes	118
Exit points for security installation exits	118
Security installation exits	118
Purpose and use of the exits	118
Environment of the exits	119
Installing the exits	120
Input	121
Return codes	122
Key generator utility program installation exit	122
Purpose and use of the exit.	122
Environment of the exit	123
Installing the exit.	124
Input	124
The SET statement.	132
Return codes	132

Chapter 6. Installation-Defined Callable Services	133
Writing a callable service	133
Contents of Registers	134
Security access control checking	135
Checking the parameters	135
Link-Editing the callable service	135
Defining a callable service	135
Writing a service stub	136
Example of a Service Stub	137
Chapter 7. Migration from PCF to z/OS ICSF	143
Running PCF and z/OS ICSF on the same system	143
Running in Compatibility Mode	144
Running in Coexistence Mode	144
Changing the master key in compatibility or coexistence mode	145
Running in noncompatibility mode	146
Specifying compatibility modes during migration	146
Converting a PCF CKDS to ICSF format	147
How the PCF conversion program runs	147
Using the conversion program override file	149
Running the Conversion Program	154
Chapter 8. Compatibility and Coexistence of 4753-HSP and ICSF	161
Running 4753-HSP and ICSF on the same z/OS system	161
Appendix A. Diagnosis Reference Information	163
Cryptographic Key Data Set (CKDS) Format	163
Format of the CKDS Header Record	163
Format of the CKDS Record	164
Format of the AES Internal Key Token	165
DES External Key Token	168
DES Null Key Token	169
Public Key Data Set (PKDS) Format	169
Format of the PKDS Header Record	169
Format of the PKDS Record	170
PKA Token Formats	170
Internal PKA Tokens	177
Token data set (TKDS) format	192
Format of the header record of the token data set	192
Format of the token and object records	193
Data Areas	200
The Cryptographic Communication Vector Table (CCVT)	200
The Cryptographic Communication Vector Table Extension (CCVE)	207
Appendix B. CICS-ICSF Attachment Facility	215
Installing the CICS-ICSF Attachment Facility	215
Steps for installing the CICS-ICSF attachment facility	215
Appendix C. Helpful Hints for ICSF First Time Startup	221
Checklist for First-Time Startup of ICSF	221
Step 1. Hardware Setup - CCF Systems	221
Step 1. Hardware Setup - PCIXCC/CEX2C Systems	221
Step 2. LPAR Activation Profiles - CCF Systems	222
Step 2. LPAR Activation Profiles - PCIXCC/CEX2C Systems	222
Step 3. ICSF Setup	223
Step 4. TKE Setup	224

Step 5. ICSF Startup	224
Step 6. Loading Master Keys and Initializing the CKDS through ICSF Panels	224
Step 7. Customizing TKE and Loading Master Keys	226
Step 8. CICS-ICSF Attachment Facility Setup	228
Step 9. Complete ICSF initialization	229
Commonly Encountered ICSF First Time Setup/initialization Messages	229
Appendix D. Using AMS REPRO Encryption	231
Steps for setting up ICSF	231
Appendix E. z890, z990, z9 EC, z9 BC and System z10 Enterprise Class without a PCIxCC or CEX2CC	233
Applications and programs	233
Callable services.	233
ICSF Setup and Initialization	234
Secure Sockets Layer (SSL)	235
TKE workstation	235
Appendix F. Accessibility	237
Using assistive technologies	237
Keyboard navigation of the user interface.	237
z/OS information	237
Notices	239
Programming Interface Information	240
Trademarks.	240
Index	243

Figures

1. The z/OS ICSF Library	xvi
2. Two Crypto PCICAs on a Processor Complex Running in LPAR Mode	60
3. Multiple Crypto Coprocessors on a Complex Running in LPAR Mode	61
4. Two Crypto CPs on a Processor Complex Running in Single Image Mode.	62
5. Two Crypto Coprocessors and one PCICC on a Processor Complex Running in LPAR Mode	63
6. Primary Panel	64
7. Coprocessor Management Panel	65
8. EXPB Control Block for Mainline Exits	94
9. EXPB Control Block in the Service Exits.	104
10. Example of a Service Entry and Exit	134
11. Example of a Service Stub.	137
12. Example of a Conversion Initial Activity Report	157
13. Example of a Conversion Update Activity Report.	159

Tables

1.	z/OS ICSF FMIDs	6
2.	FMID and Hardware	6
3.	Exit Identifiers and Exit Invocations	35
4.	Summary of new and changed ICSF callable services	44
5.	IPCS Symbols and Format References for the ICSF Control Blocks	85
6.	EXPB Control Block Format for Mainline Exits	94
7.	CSFEXIT1 Parameters	95
8.	CSFEXIT2 and CSFEXIT3 Parameters.	96
9.	CSFEXIT4 and CSFEXIT5 Parameters.	97
10.	Format of the Exit Name Table.	97
11.	Services and Their ICSF Names	101
12.	Compatibility Services and Their ICSF Names	103
13.	EXPB Control Block Format for Services	104
14.	SPB Control Block Format	107
15.	The CKDS Entry Retrieval Exit Parameters.	111
16.	CVXP Control Block Format	113
17.	RWXP Control Block Format	117
18.	Parameters Received by the Security Service Exit	121
19.	Parameters Received by the Security Key Exit	121
20.	KGXP Control Block Format	124
21.	Format of Records in the Override File	150
22.	Cryptographic Key Data Set Header Record Format	163
23.	Cryptographic Key Data Set Record Format	164
24.	Internal Key Token Format	165
25.	Internal Key Token Format	166
26.	Internal Clear Key Token Format	167
27.	Format of External Key Tokens	168
28.	Format of Null Key Tokens	169
29.	Public Key Data Set Header Record Format	169
30.	Public Key Data Set Record Format	170
31.	RSA Public Key Token	171
32.	DSS Public Key Token	171
33.	RSA Private External Key Token Basic Record Format	172
34.	RSA Private Key Token, 1024-bit Modulus-Exponent External Format	173
35.	RSA Private Key Token, 4096-bit Chinese Remainder Theorem External Format	174
36.	DSS Private External Key Token	176
37.	RSA Private Internal Key Token Basic Record Format.	177
38.	RSA Private Internal Key Token, 1024-bit ME Form for Cryptographic Coprocessor Feature	179
39.	RSA Private Internal Key Token, 1024-bit ME Form for PCICC, PCIXCC or CEX2C	179
40.	RSA Private Internal Key Token, 4096-bit Chinese Remainder Theorem Internal Format	181
41.	DSS Private Internal Key Token	182
42.	Trusted Block Token	184
43.	Protection Information TLV Object	187
44.	Activation and Expiration Data TLV Object	187
45.	Transport Key Variant TLV Object	188
46.	Transport Key Rule Reference TLV Object	188
47.	Common Export Key Parameters TLV Object	189
48.	Export Key Rule Reference TLV Object	190
49.	CCA Token Export Key Parameters TLV Object	191
50.	Format of the header record of the token data set	192
51.	Format of the common section of the token and object records	193
52.	Format of the unique section of the token record	194
53.	Format of the token object flags	194

54.	Format of the token certificate object	196
55.	Format of the token public key object	196
56.	Format of the token private key object	197
57.	Format of the token secret key object.	198
58.	Format of the token data object	199
59.	Cryptographic Communication Vector Table	201
60.	Cryptographic Communication Vector Table Extension.	207
61.	Master Key Verification Pattern Block Format	211
62.	Generic Service Table Block Format	212
63.	RMF Measurements Record Format	212

About this information

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

- z/OS Integrated Cryptographic Service Facility (ICSF)
- z/OS Open Cryptographic Services Facility (OCSF)
- z/OS System Secure Socket Level Programming (SSL)
- z/OS Public Key Infrastructure Services (PKI)

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 information

This information is intended for the system programmer. It describes the 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 information

This information is divided into descriptions of these tasks:

- Introducing ICSF
 - Chapter 1, “Introduction to z/OS ICSF,” on page 1, introduces the cryptographic key data set (CKDS), the public key data set (PKDS) and the token data set (TKDS) 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 13, describes how to customize SYS1.PARMLIB, create the CKDS, the PKDS, and the TKDS, 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.

- Migration and coexistence issues

Chapter 7, “Migration from PCF to z/OS ICSF,” on page 143, describes how to migrate application programs and cryptographic key data set information to z/OS ICSF from the IBM cryptographic products CUSP/PCF.

Chapter 3, “Migration,” on page 43, describes migration to z/OS ICSF from previous releases of ICSF.

Chapter 8, “Compatibility and Coexistence of 4753-HSP and ICSF,” on page 161, gives a brief overview of migrating 4753-HSP key storage to ICSF CKDS.

- Customizing ICSF

- Chapter 6, “Installation-Defined Callable Services,” on page 133 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 5, “Installation Exits,” on page 87, describes the ICSF installation exits you can use to customize ICSF.

- Operating ICSF

- Chapter 4, “Operating ICSF,” on page 57, describes how to add and remove cryptographic coprocessors and to start, modify, and stop ICSF and other operating considerations.

- “Event Recording” on page 70, describes ICSF event recording on the Security Console and SMF.

- Planning ICSF

- “Security Considerations” on page 77, describes methods you can use to protect ICSF resources.

- Diagnosing ICSF

- “Debugging Aids” on page 79, describes the use of component trace and Interactive Problem Control System (IPCS) to debug ICSF.

- Appendix A, “Diagnosis Reference Information,” on page 163, 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, “CICS-ICSF Attachment Facility,” on page 215, defines steps to install the CICS-ICSF Attachment Facility.

- Appendix C, “Helpful Hints for ICSF First Time Startup,” on page 221, defines helpful hints and that you may encounter when starting ICSF for the first time.

- Appendix D, “Using AMS REPRO Encryption,” on page 231, provides information on using IDCAMS REPRO ENCIPHER and DECIPHER options with ICSF.

- Appendix E, “z890, z990, z9 EC, z9 BC and System z10 Enterprise Class without a PCIXCC or CEX2CC,” on page 233 describes processing and functionality support for this environment.

- Appendix F, “Accessibility,” on page 237 contains information on accessibility features in z/OS.

- “Notices” on page 239 contains information on notices, programming interface information and trademarks.

Where to find more information

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

This publication also refers to these publications:

- *IBM ES/3090 Processor Complex Recovery Guide*
- *z/OS 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
- *z/OS DFSMS Access Method Services for Catalogs*
- *z/OS DFSMS Using Data Sets*
- *IBM Transaction Security System: General Information Manual and Planning Guide*
- *IBM Transaction Security System: Concepts and Programming Guide: Volume 1, Access Controls and DES Cryptography*
- *IBM Transaction Security System: Basic CCA Cryptographic Services*
- *IBM Transaction Security System: Concepts and Programming Guide: Volume II, Public-Key Cryptography*
- *IBM Distributed Key Management System, Installation and Customization Guide*
- *OS/VS1 and OS/VS2 MVS Cryptographic Unit Support: Installation Manual*
- *OS/VS1 and OS/VS2 MVS Programmed Cryptographic Facility*
- *CICS Customization Guide*, SC34-6227
- *CICS Resource Definition Guide*, SC34-6228

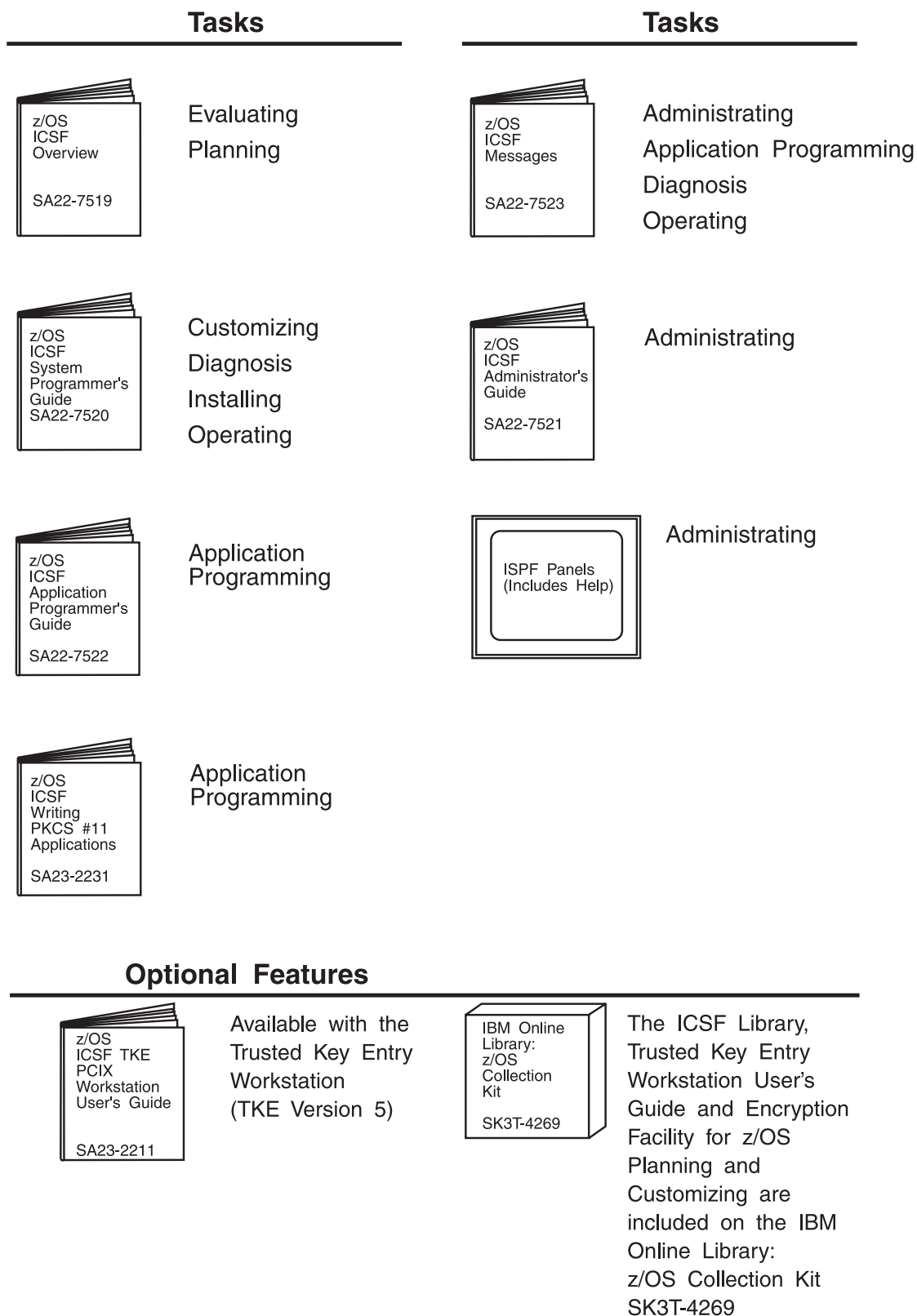


Figure 1. The z/OS ICSF Library

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.

Summary of changes

Summary of changes for SA22-7520-12 z/OS Version 1 Release 9

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-011, which supports z/OS Version 1 Release 9.

This release of ICSF, HCR7750, runs on z/OS V1R7, V1R8 and V1R9 and only on zSeries hardware.

New information

- Added support for new server hardware: System z10 Enterprise Class
- Added new support for CPACF

Changed information

- These callable services have been changed to support new SHA-2 algorithms:
 - CSNBOWH, CSNBOWH1, CSNEOWH and CSNEOWH1- One-Way Hash Generate
 - CSFIQF and CSFIQF6 - ICSF Query Service

Summary of changes for SA22-7520-11 z/OS Version 1 Release 9

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-010, which supports z/OS Version 1 Release 9.

This release of ICSF, HCR7750, runs on z/OS V1R7, V1R8 and V1R9 and only on zSeries hardware.

New information

- Added support for 4096-bit RSA keys on
- Added support for ISO-3 PIN block format

Changed information

- These callable services have been changed to support 4096-bit RSA keys:
 - CSNDPKD and CSNFPKD - PKA Decrypt
 - CSNDPKE and CSNFPKE - PKA Encrypt
 - CSNDPKX and CSNFPKX - PKA Key Extract
 - CSNDPKG and CSNFPKG - PKA Key Generate
 - CSNDPKI and CSNFPKI - PKA Key Import
 - CSNDPKB and CSNFPKB - PKA Key Token Build
 - CSNDKTC - PKA Key Token Change
 - CSNDKRC and CSNFKRC - PKDS Record Create
 - CSNDKRW - PKDS Record Write
 - CSNDKRD and CSNFKRD - PKDS Record Delete
 - CSNDRKX - Remote Key Export
 - CSNDSYX - Symmetric Key Export

- CSNDSYG - Symmetric Key Generate
- CSNDSYI - Symmetric Key Import
- CSNDTBC - Trusted Block Create
- CSNDDSG - Digital Signature Generate
- CSNDDSV - Digital Signature Verify
- These callable services have been changed to support ISO-3 PIN block format:
 - CSNBCPE - Clear PIN Encrypt
 - CSNBEPG - Encrypted PIN Generate
 - CSNBPTR - Encrypted PIN Translate
 - CSNBPVR - Encrypted PIN Verify
 - CSNBPCU - PIN Change/Unblock
 - CSNBSPN - Secure Messaging for PINs
- The callable service, Random Number Generate Long (CSNBRNGL and CSNERNGL), now allows a user to specify the length when generating a random number.

This document has been enabled for the following types of advanced searches in the online z/OS Library Center: examples, tasks, concepts, references, and parmlib members.

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.

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.

Summary of changes for SA22-7520-10 z/OS Version 1 Release 9

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-09, which supports z/OS Version 1 Release 8.

This release of ICSF, HCR7740, only runs on z/OS V1R9 and only on zSeries hardware.

New information

- These callable services have been added to support PKCS #11 token management:
 - CSFPTRC - Token Record Create
 - CSFPTRD - Token Record Delete
 - CSFPTRL - Token Record List
 - CSFPSAV - Set Attribute Value
 - CSFPGAV - Get Attribute Value

Changed information

- These callable services have been changed to support Cipher Feedback Mode (CFB) and PKCS #7 padding for encryption:
 - CSNBSYD - Symmetric Key Decipher (new CFB and PKCS-PAD keywords)
 - CSNBSYE - Symmetric Key Encipher (new CFB and PKCS-PAD keywords)
- Changed TALENGTH(140) to TALENGTH (250) of using a CICS Command Level Interpreter Transaction (CECI) to enable CSFATRUE in Steps for installing the CICS-ICSF attachment facility.
- Added note stating because it is possible for ICSF control blocks like the DACC and CCVT to persist in storage across an ICSF restart, an IPL is required when installing a new release of ICSF.
- Added Dispatching Priority section. In order to avoid performance problems, the dispatching priority of ICSF should be set at least as high as that of the highest task using ICSF.
- Added information about logging in SMF the results of any authorization checks that are made.
- Added improvements on auditing ICSF events.

This document has been enabled for the following types of advanced searches in the online z/OS Library Center: examples, tasks, concepts, references, and parmlib members.

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.

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.

Summary of changes for SA22-7520-09 z/OS Version 1 Release 8

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-08, which supports z/OS Version 1 Release 7.

This release of ICSF, HCR7731, only runs on z/OS or z/OS.e V1R6/V1R7 and only on zSeries hardware.

New information

- The following callable services have been added to support remote key load:
 - CSNDTBC - Trusted Block Create
 - CSNDRKX - Remote Key Export

Changed information

- The following Callable Services have been changed to support remote key load:
 - CSNDPKI - PKA Key Import
 - CSNDDSV - Digital Signature Verify
 - CSNDKTC - PKA Key Token Change
- Added Enhanced PIN Security for the following callable services:

- Clear PIN Encrypt - CSNBCPE
- Clear PIN Generate Alternate - CSNBCPA
- Encrypted PIN Generate - CSNBEPG
- Encrypted PIN Translate - CSNBPTR
 - Encrypted PIN Translate enhanced further to produce return code 8,3016 - The value of the PAD data is not valid.
- Encrypted PIN Verify - CSNBPVR
- PIN Change/Unblock - CSNBPCU
- Added support for ISO 16609 CBC mode T-DES - New rule array keyword, TDES-MAC, will be added to the MAC Generate and MAC Verify Services.

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.

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 SA22-7520-08 z/OS Version 1 Release 7

This document contains information previously presented in *z/OS ICSF System Programmer's Guide*, SA22-7520-07, which supports z/OS Version 1 Release 6.

ICSF 64-bit Virtual Support (HCR7720) only runs on z/OS or z/OS.e V1R6 and V1R7 and only on zSeries hardware.

This release of ICSF, HCR7730, only runs on z/OS or z/OS.e V1R6 and V1R7 and only on zSeries hardware.

New information

- Added support for new server hardware
- Added support for CryptoExpress2 Accelerator
- Added key management support for AES; clear key tokens only
- Added support for sysplex-wide consistency of CKDS

Changed information

- Added support for CPACF hardware
- The following Callable Services have been changed:
 - CSNBOWH, CSNBOWH1 - one way hash - enhanced to support SHA-256 algorithm.
 - CSNBSYD, CSNBSYD1 - symmetric key decipher - enhanced to use hardware for AES encryption; only 128-bit keys supported; only available with IBM System z9 109
 - CSNBSYE, CSNBSYE1 - symmetric key encipher - enhanced to use hardware for AES encryption; only 128-bit keys supported; only available with IBM System z9 109

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.

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.

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

This topic describes the cryptographic hardware features available. Information on adding and removing cryptographic coprocessors can be found in *z/OS Cryptographic Services ICSF Administrator's Guide*.

Cryptographic Hardware

This topic describes the cryptographic hardware features available. Information on adding and removing cryptographic coprocessors can be found in *z/OS Cryptographic Services ICSF Administrator's Guide*.

Crypto Express2 Feature

The Crypto Express2 Feature is an asynchronous cryptographic coprocessor or accelerator. The feature can be configured by an installation as an accelerator or coprocessor. It is available on the IBM System z9 Enterprise Class, IBM System z9 Business Class, and System z10 Enterprise Class.

The Crypto Express2 Coprocessor provides function equivalent to the PCIXCC. The Crypto Express2 Accelerator provides function equivalent to the PCICA with significantly enhanced performance.

PCI X Cryptographic Coprocessor (PCIXCC) and Crypto Express2 Coprocessor (CEX2C)

The PCI X Cryptographic Coprocessor is an asynchronous cryptographic coprocessor. It is a replacement for the Cryptographic Coprocessor Feature and PCI Cryptographic Coprocessor. It is only available on a IBM @server zSeries 990 or IBM @server zSeries 890.

The Crypto Express2 Coprocessor (CEX2C) provides equivalent function to the PCIXCC, but is packaged differently.

The PCIXCC/CEX2C symmetric-keys master key is used in place of the CCF DES master key. The asymmetric-keys master key is used in place of the CCF signature and key management master keys. The PCIXCC/CEX2C supports up to 2048-bit RSA keys in all PKA services except SET services (Set Block Compose and Set Block Decompose).

HCR7750 supports 4096-bit RSA keys, but only on a Crypto Express2 Coprocessor (CEX2C).

This feature is in the process of being certified for Federal Information Processing Standard (FIPS) 140-2. This includes algorithmic certification under FIPS 46-2 (DES) and FIPS 180-1 (Secure Hash Standard).

CP Assist for Cryptographic Functions (CPACF)

CPACF is a set of cryptographic instructions available on all CPs of a z990, z890, z9 EC, z9 BC and System z10 Enterprise Class. Use of the CPACF instructions provides improved performance. The SHA-1 algorithm is always available. Additionally, SHA-224 and SHA-256 algorithms are available on the z9 EC and z9 BC. Additionally, SHA-384 and SHA-512 algorithms are available on System z10 Enterprise Class.

CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement, feature 3863, provides for clear key DES and TDES instructions. On the z9 EC and z9 BC, this feature includes clear key AES for 128-bit keys. On System z10 Enterprise Class, this feature also includes AES for 192-bit and 256-bit keys.

If you want to include a PCIXCC, CEX2C, PCICA (z990, z890) or Crypto Express2 feature (z9 EC, z9 BC and System z10 Enterprise Class), then feature 3863 is required.

PCI Cryptographic Accelerator (PCICA)

On all systems, the PCI Cryptographic Accelerator provides support for clear keys in the CSNDPKD callable services for better performance than when executed in a cryptographic coprocessor. On z990 or z890, it also supports CSNDDSV and CSNDPKE.

PCICAs enable maximum SSL performance.

Cryptographic Coprocessor Feature (CCF)

The Cryptographic Coprocessor Feature (CCF) can have up to two cryptographic coprocessors as high-speed extensions of the central processor. Each CCF contains both DES and PKA cryptographic processing units. You can configure the processor complex to run in either single-image mode or logical partition mode.

If the Cryptographic Coprocessor Feature is in single-image mode, the same master keys must be installed on both CCFs. If you bring a second coprocessor online, ICSF verifies that the master keys are the same. If the DES master keys are different, ICSF will not use the second coprocessor. The PKA master keys must be the same on both Coprocessors in order to enable the PKA services.

This feature is currently certified for Federal Information Processing Standard (FIPS) 140-1 level 4. This includes algorithmic certification under FIPS 46-2 (DES), FIPS 180-1 (Secure Hash Standard), and FIPS 186 (Digital Signature Standard).

The possible configurations include:

- DES with PKA
These servers are configured for full 64-bit DES keys (effective length 56 bits), 1024-bit PKA keys for DES key distribution, and 1024-bit PKA signature keys.
- Triple DES with PKA

These servers are configured for 192-bit DES keys (effective length 169 bits), 1024-bit PKA keys for DES key distribution, and 1024-bit PKA signature keys. This configuration is available on S/390 G5 Enterprise Servers and higher.

PCI Cryptographic Coprocessor

The PCI Cryptographic Coprocessor, which works in conjunction with the Cryptographic Coprocessor Feature, provides the capability of generating and retaining RSA keys in secure hardware. This capability meets a requirement to become a SET Certificate Authority. A PCI Cryptographic Coprocessor is required on a CCF system for:

- UDX capability
- Generating RSA public and private keys
- The retained key list and retain key delete callable service.
- See “Callable Services” on page 44 for other functions requiring a PCICC

The PCICC cards are in addition to the Cryptographic Coprocessor Feature. In order for the PCI Cryptographic Coprocessor to operate, the verification pattern for the SYM-MK master key must match the verification pattern of the DES master key on the server’s Cryptographic Coprocessor Feature. Before you can use the PKA services of the PCI Cryptographic Coprocessor, you must install both the KMMK and the SMK on the Cryptographic Coprocessor Feature and the ASYM-MK master key on the PCI Cryptographic Coprocessor. The hash pattern of the ASYM-MK master key must match the hash pattern of the SMK in order to use the PCI Cryptographic Coprocessor.

Note: For new installations, it is recommended that the installation enter the KMMK equal to the SMK master key. Existing customers should reencipher their PKDS and migrate to a system with the KMMK equal to the SMK.

Server Hardware

This topic describes the servers on which the cryptographic hardware features are available.

System z10 Enterprise Class

The System z10 Enterprise Class provides constraint relief and addresses various customer demands. It has several cryptographic features.

- **CP Assist for Cryptographic Functions** is implemented on every processor. SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 secure hashing is directly available to application programs.
- Feature code 3863, **CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement** – enables clear key DES and TDES instructions on all CPs. AES 128-bit, AES 192-bit and AES 256-bit support is also available.
- Feature code 0863, **Crypto Express2 Feature** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The System z10 Enterprise Class can support a maximum of 8 features. Each feature code has two coprocessors/accelerators.

IBM System z9 Business Class (z9 BC)

The IBM System z9 BC provides constraint relief and addresses various customer demands. It has several cryptographic features.

- **CP Assist for Cryptographic Functions** is implemented on every processor. SHA-1, SHA-224 and SHA-256 secure hashing is directly available to application programs.
- Feature code 3863, **CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement** – enables clear key DES and TDES instructions on all CPs. In addition, ICSF supports hardware implementation of AES 128-bit keys and software implementation of AES 192-bit and AES 256-bit key lengths.
- Feature code 0863, **Crypto Express2 Feature** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM System z9 BC can support a maximum of 8 features. Each feature code has two coprocessors/accelerators.

IBM System z9 Enterprise Class (z9 EC)

The IBM System z9 EC provides constraint relief and addresses various customer demands. It has several cryptographic features.

- **CP Assist for Cryptographic Functions** is implemented on every processor. SHA-1, SHA-224 and SHA-256 secure hashing is directly available to application programs.
- Feature code 3863, **CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement** – enables clear key DES and TDES instructions on all CPs. In addition, ICSF supports hardware implementation of AES 128-bit keys and software implementation of AES 192-bit and AES 256-bit key lengths.
- Feature code 0863, **Crypto Express2 Feature** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM System z9 EC can support a maximum of 8 features. Each feature code has two coprocessors/accelerators.

IBM @server zSeries 990 (z990)

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

- **CP Assist for Cryptographic Functions** is implemented on every processor. SHA-1 secure hashing is directly available to application programs.
- Feature code 3863, **CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement** – enables clear key DES and TDES instructions on all CPs. In addition, ICSF supports software implementation of AES.
- Feature code 0862, **PCI Cryptographic Accelerator** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM @server zSeries 990 can support a maximum of 12 PCI Cryptographic Accelerators. Each feature code has two coprocessors.
- Feature code 0868, **PCI X Cryptographic Coprocessor** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM @server zSeries 990 can support a maximum of 4 PCIXCCs. Each feature code has one coprocessor.
- Feature code 0863, **Crypto Express2 Coprocessor** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM @server zSeries 990 can support a maximum of 16 CEX2Cs. Each feature code has two coprocessors.

Note: You can have a maximum of 6 PCICA features (12 cards), 4 PCIXCC features (4 cards) and 16 CEX2Cs (8 features), with maximum number of 8 installed features.

IBM @server zSeries 890 (z890)

The IBM @server zSeries 890 provides constraint relief and addresses various customer demands. It has several cryptographic features.

- **CP Assist for Cryptographic Functions** are implemented on every processor. SHA-1 secure hashing is directly available to application programs.
- Feature code 3863, **CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement** – enables clear key DES and TDES instructions on all CPs. In addition, ICSF supports software implementation of AES.
- Feature code 0862, **PCI Cryptographic Accelerator** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM @server zSeries 890 can support a maximum of 12 PCI Cryptographic Accelerators. Each feature code has two coprocessors.
- Feature code 0868, **PCI X Cryptographic Coprocessor** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM @server zSeries 890 can support a maximum of 4 PCIXCCs. Each feature code has one coprocessor.
- Feature code 0863, **Crypto Express2 Coprocessor** – optional, and only available if you have feature 3863, CPACF DES/TDES Enablement installed. The IBM @server zSeries 890 can support a maximum of 16 CEX2Cs. Each feature code has two coprocessors.

Note: You can have a maximum of 6 PCICA features (12 cards), 4 PCIXCC features (4 cards) and 16 CEX2Cs (8 features), with maximum number of 8 installed features.

IBM @server zSeries 900 (z900) — Feature Code 800

You can enable these features on this server:

- **Cryptographic Coprocessor Feature** – one or two cryptographic coprocessors protected by tamper-detection circuitry and a cryptographic battery unit.
- Feature code 0861, **PCI Cryptographic Coprocessor (PCICC)** – based on the 4758 model 2 standard PCI-bus card package. You must have at least one Cryptographic Coprocessor Feature on your system with a PCICC. Note that each feature has two coprocessors.
- Feature code 0862, **PCI Cryptographic Accelerator (PCICA)**. You must have at least one Cryptographic Coprocessor Feature on your system with a PCICA. Note that each feature has two coprocessors.

Note: The 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.

IBM @server zSeries 800 (z800) — Feature Code 800

These features are available on this server:

- **Cryptographic Coprocessor Feature (CCF)** – one or two cryptographic coprocessors protected by tamper-detection circuitry and a cryptographic battery unit.
- Feature code 0861, **PCI Cryptographic Coprocessor (PCICC)** – based on the 4758 model 2 standard PCI-bus card package. You must have at least one Cryptographic Coprocessor Feature on your system with a PCICC. Note that each feature has two coprocessors.

- Feature code 0862, **PCI Cryptographic Accelerator (PCICA)**. You must have at least one Cryptographic Coprocessor Feature on your system with a PCICA. Note that each feature code has two coprocessors.

Note: The IBM @server zSeries 800 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.

z/OS ICSF FMIDs

These tables explain the relationships of z/OS releases, ICSF FMIDs and servers.

Table 1. z/OS ICSF FMIDs

z/OS and z/OS.e	ICSF FMID ¹	Web deliverable name
V1.7	HCR7720	ICSF 64-bit Virtual Support for z/OS V1.6 and z/OS.e V1.6 (included in base)
	HCR7730	Cryptographic support for z/OS V1R6/V1R7 and z/OS.e V1R6/V1R7 ²
	HCR7731	Enhancements for Cryptographic support for z/OS and z/OS.e V1R6/R7
	HCR7750	Cryptographic support for z/OS V1R7-R9 and z/OS.e V1R7-R8.
V1.8	HCR7730	Cryptographic support for z/OS V1R6/V1R7 and z/OS.e V1R6/V1R7 ²
	HCR7731	Enhancements for Cryptographic support for z/OS and z/OS.e V1R6/R7
	HCR7750	Cryptographic support for z/OS V1R7-R9 and z/OS.e V1R7-R8.
V1.9	HCR7740	Enhancements for Cryptographic support for z/OS V1R9 (included in base)
	HCR7750	Cryptographic support for z/OS V1R7-R9 and z/OS.e V1R7-R8.
Notes: 1. PTF information can be found in the PSP bucket '2094DEVICE'. 2. No longer downloadable		

Refer to this chart to determine what release is associated with each ICSF FMID and what server it will run on.

Table 2. FMID and Hardware

ICSF FMID	Applicable z/OS Releases	Servers where FMID will run
HCR7720 (WD#4 - Base of z/OS 1.7)	1.7	z800, z900, z890, z990, z9 EC, z9 BC and System z10 Enterprise Class
HCR7730 (WD#5)	1.7, 1.8	z800, z900, z890, z990, z9 EC, z9 BC and System z10 Enterprise Class
HCR7731 (WD#6 - Base of z/OS 1.8)	1.7, 1.8	z800, z900, z890, z990, z9 EC, z9 BC and System z10 Enterprise Class
HCR7740 (Base of z/OS 1.9 only)	1.9	z800, z900, z890, z990, z9 EC, z9 BC and System z10 Enterprise Class
HCR7750	1.7, 1.8 and 1.9	z800, z900, z890, z990, z9 EC, z9 BC and System z10 Enterprise Class

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, Advanced Encryption Standard (AES), or the Commercial Data Masking Facility (CDMF).

Restriction: The CDMF defines a scrambling technique for data confidentiality. It is a weakened form of DES and is only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900 servers.

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.)

Restriction: DSS is only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900 servers.

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, PCI X Cryptographic Coprocessor or Crypto Express2 Coprocessor. On CCF systems, the SYM-MK must be the same as the DES master key on the Cryptographic Coprocessor Feature. On systems with PCI X Cryptographic Coprocessors and Crypto Express2 Coprocessors, the SYM-MK hash pattern must match the hash pattern of the CKDS.

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 PCICC or PCIXCC/CEX2C is a triple-length key used to encipher and decipher PKA keys. In order for the PCI Cryptographic Coprocessor to function, the value of the ASYM-MK must have the same value as the SMK on the Cryptographic Coprocessor Feature. If the PCICC master key values are different, then the PCICC will not be made active. On systems with PCIXCCs/CEX2Cs, the ASYM-MK hash pattern must match the hash pattern of the PKDS.

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.

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 Cryptographic Services 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 z800 or z900, you can:

- Use KGUP to generate or enter keys
- Use the dynamic CKDS update callable services to create and write keys directly to the CKDS
- 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.

To add operational keys to the CKDS for the z890, z990, z9 EC, z9 BC and System z10 Enterprise Class servers, you can:

- Use KGUP to generate or enter keys or to load keys from the PCIXCC/CEX2C key part registers
- Use the dynamic CKDS update callable services to create and write keys directly to the CKDS
- Use the Trusted Key Entry workstation (TKE Version 4.1 or higher) to load operational DES keys. TKE is not part of the base product. It is an optional feature.

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.

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 must be initialized.

Note: If a trusted block token is put into the PKDS by an HCR7731 ICSF system, a prior release of ICSF will not be able to reencipher the PKDS. When the prior release of ICSF comes across a trusted block token, encipher will be terminated and a message will be issued indicating that some tokens were reenciphered.

Support to REENCIPHER PKDS and ACTIVATE PKDS is available by using the Master Key Management Panels or 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 Cryptographic Services ICSF Administrator's Guide*, SA22-7521.

The Token Data Set (TKDS)

PKCS #11 tokens and objects are stored in a VSAM data set called the token data set (TKDS). ICSF provides a sample TKDS allocation job (member CSFTKDS) in SYS1.SAMPLIB. The TKDS contains individual entries for each token and object that is added to it. ICSF maintains two copies of the TKDS: a disk copy and an in-storage copy. Only token objects are stored in the TKDS, session objects are stored in a data space.

The TKDS must be a key-sequenced data set with spanned variable length records and must be allocated on a permanently resident volume.

Additional Background Information

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

Running PCF applications on z/OS 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:

- 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 161.

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 changes the status of a cryptographic 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, PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor or Crypto Express2 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 suppress 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). These measurements are of the Direct Access Crypto CCF instructions and the PCIXCCs and CEX2Cs.

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.

Steps prior to starting installation

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

When beginning installation:

1. Refer to *z/OS 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 Server (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 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.

Chapter 2. Installation, Initialization, and Customization

For this topic, you need to understand these 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 pass phrase. 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 for 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 publication for details. This publication explains these installation steps.

Note: Because it is possible for ICSF control blocks like the DACC and CCVT to persist in storage across an ICSF restart, an IPL is required when installing a new release of ICSF.

1. Customize SYS1.PARMLIB. "Steps to customize SYS1.PARMLIB" on page 14 describes this task.
2. Create the Cryptographic Key Data Set (CKDS). "Steps to create the CKDS" on page 15 describes this. Create the Public Key Data Set (PKDS). "Steps to create the PKDS" on page 16 describes this task.
3. If PKCS #11 support is desired, create the TKDS. "Steps to create the TKDS" on page 19 describes this task.
4. Create the installation options data set. "Steps to create the Installation Options Data Set" on page 21 describes this task.
5. Create the startup procedure. "Steps to create the ICSF Startup Procedure" on page 23 describes this task.
6. Provide access to the ICSF panels. "Steps to provide access to the ICSF panels" on page 24 describes this task.

Note: You only need to perform the first six steps once. If you stop ICSF and want to perform a subsequent SMP/E electronic delivery (this is optional), you need to start at Step 7 (Start ICSF for the first time).

7. Start ICSF for the first time. See “Steps to start ICSF for the first time” on page 26. Once ICSF has been started, Master Keys can be entered.

For additional information on ICSF first time startup, refer to “Checklist for First-Time Startup of ICSF” on page 221. See *z/OS Cryptographic Services ICSF Administrator's Guide* for directions on entering Master Keys.

8. Enter Master Keys.

9. Run the JCL to set the SMP/E pass phrase for SMP/E electronic delivery (optional). “MK Initialization for SMP/E - CCF Systems Only” on page 30 describes this.

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

Other topics in this publication and *z/OS Cryptographic Services ICSF Administrator's Guide* provide additional installation information.

For information on installing the CICS-ICSF Attachment Facility, refer to Appendix B, “CICS-ICSF Attachment Facility,” on page 215.

Steps to 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 LNKLIST concatenation. This adds the ICSF library to the z/OS library search. This is an example of an ICSF entry to the LNKLIST concatenation.

```
CSF.SCSFMOD0
```

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

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

3. In the IKJTSOxx parameter, add CSFDAUTH and CSFDPKDS as a value in the AUTHPGM parameter list and in the AUTHTSF parameter list. This 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              */ +
CSFDPKDS              /* ICSF              */ +
```

4. If you will be using TKE V3.0 or higher 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.

Steps to 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.

Attention: Ensure that this volume is not subject to data set migration. If the CKDS is migrated, message CSFM450E is issued and ICSF ends.

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 (CCF systems only). The other system keys, NOCV enablement keys and ANSI enablement keys, are optional.

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

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 S/390 Enterprise Servers and S/390 Multiprise and IBM @server zSeries 900, there may also be three enhanced 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 *z/OS DFSMS Access Method Services for Catalogs*.

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 *z/OS DFSMS Access Method Services for Catalogs*.

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:

- 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 this SYS1.SAMPLIB CSFCKDS member sample:

```
//CSFCKDS JOB = JOB CARD PARAMETERS
//*****
//* Licensed Materials - Property of IBM
//* 5694-A01
//* (C) Copyright IBM Corp. 2002
//*
//* 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) Change xxxxxx to the valid where you want your CKDS to
//*    reside. 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)) -
        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 *z/OS DFSMS Access Method Services for Catalogs*.

Steps to create the PKDS

The PKDS must be allocated and the PKDS data set name must be specified on the PKDSN parameter of the options data set when you first start ICSF. ICSF support for the PCI Cryptographic Coprocessor or PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor requires a PKDS. Even if not available at first time start up, a PCICC, PCIXCC or CEX2C can be dynamically configured online. Since ICSF can not tell if a PCICC, PCIXCC or CEX2C 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 *z/OS DFSMS Access Method Services for Catalogs*.
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 *z/OS DFSMS Access Method Services for Catalogs*.
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:

- 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 Server (RACF) data set profile for the PKDS.
 - The CISZ(8192) coded in this sample in the DATA section is a hardcoded requirement.
4. Allocate a disk copy of the PKDS by defining a VSAM cluster as in this SYS1.SAMPLIB CSFPKDS member sample:

```
//CSFPKDS JOB = JOB CARD PARAMETERS
//*****
//* Licensed Materials - Property of IBM                      *
//* 5694-A01                                                  *
//* (C) Copyright IBM Corp. 2002, 2007                      *
//*
```

```

/* THIS JCL DEFINES A VSAM PKDS 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) Change xxxxxx to the valid where you want your PKDS to        *
/*    reside. 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(350,3800) -
        KEYS(72 0) -
        FREESPACE(0,0) -
        SHAREOPTIONS(2,3)) -
    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 *z/OS DFSMS Access Method Services for Catalogs*.

Migrating to a larger PKDS

For FMID HCR7750, the LRECL for the PKDS has changed and customers must migrate to a new PKDS. The PKDS must be migrated before the customers can start HCR7750.

To create a larger PKDS, use the JCL in SYS1.SAMPLIB(CSFPKDS).

If the PKDS is shared with down-level systems, install the toleration APAR on those systems to allow continued sharing of the PKDS. ICSF will be able to interact with both sizes of the PKDS. The toleration APAR number is OA21807.

The steps to migrate are:

1. Make a backup copy of the PKDS you are currently using.
2. Create a larger PKDS - use the JCL in SYS1.SAMPLIB(CSFPKDS) from the HCR7750 system. If the PKDS will be shared, place the VSAM dataset where it can be shared.
3. Copy the old PKDS to the larger PKDS using the JCL in SYS1.SAMPLIB(CSFPKDCP) from the HCR7750 system. Protect the VSAM data sets from viewing by non-authorized personnel.

```

//CSFPKDCP <JOB CARD PARAMETERS>
//*****
/* Licensed Materials - Property of IBM                      *
/* 5694-A01                                                    *
/* Copyright IBM Corp. 2007                                    *

```

```

/*
/* THIS JCL COPIES ONE VSAM PKDS TO THE LARGER PKDS
/*
/* 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.
/*
/******
//STEP1 EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//INDD DD DSN=CSF.CSF PKDS.OLD,DISP=SHR
//OUTDD DD DSN=CSF.CSF PKDS,DISP=SHR
//SYSIN DD *
REPRO INFILE(INDD) OUTFILE(OUTDD)
/*

```

4. Update the ICSF started procedures on all systems to reference the new PKDS. If the new PKDS was empty when ICSF was started, you must initialize it before PKA callable services can be enabled.
5. Stop and restart ICSF on each system. If the system has not been IPLed since HCR7750 was installed, then an IPL is required.

Steps to create the TKDS

If PKCS #11 support is desired then the TKDS must be allocated and the TKDS data set name must be specified on the TKDSN parameter of the options data set when you first start ICSF.

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

1. Determine the amount of primary space you need to allocate for the TKDS.
This should reflect the total number of entries you expect the data set to contain originally. The TKDS will contain PKCS #11 tokens and objects. Each record has a maximum size of 32 KB. A record for a token will use 0.1 KB. The minimum size of a record for objects is: Data: 1 KB, Secret Key: 1.1 KB, Public Key: 1.5 KB, Private Key: 3.4 KB, Certificate: 1 KB. Allocate enough space for the number of tokens to be supported and for the number of objects to be created. Note that session objects are not stored in the TKDS.

2. Determine the amount of secondary space to allocate for the TKDS.
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 *z/OS DFSMS Access Method Services for Catalogs*.

To access tokens and objects, VSAM uses the token handle or object handle as the VSAM key. This means that VSAM adds objects to the data set in collating sequence. That is, if two objects named A and B are in the data set, A appears earlier in the data set than B. As a result, adding objects 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 objects 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 TKDS 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 *z/OS DFSMS Access Method Services for Catalogs*.

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

Note: To improve security and reliability of the data that is stored on the TKDS:

- Use the **ERASE** and **WRITECHECK** parameters on the **AMS DEFINE CLUSTER** command. **ERASE** overwrites data records with binary zeros when the TKDS 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 TKDS.

4. Allocate a disk copy of the TKDS by defining a VSAM cluster as in this **SYS1.SAMPLIB CSFTKDS** member sample:

```
//CSFTKDS JOB = JOB CARD PARAMETERS
//*****
/* Licensed Materials - Property of IBM                      *
/* 5694-A01                                                    *
/* (C) Copyright IBM Corp. 2007                               *
/*                                                              *
/* THIS JCL DEFINES A VSAM TKDS 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) Change XXXXXX to the volid where you want your TKDS to   *
/*    reside. The TKDS needs to be on a permanently resident   *
/*    volume.                                                  *
/*                                                              *
//*****
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER (NAME(CSF.CSFTKDS) -
        VOLUMES(XXXXXX) -
        RECORDS(100,50) -
        RECORDSIZE(2200 32756) -
        KEYS(72 0) -
        FREESPACE(0,0) -
        SPANNED -
        SHAREOPTIONS(2,3)) -
    DATA (NAME(CSF.CSFTKDS.DATA) -
        BUFFERSPACE(100000) -
        ERASE -
        WRITECHECK) -
    INDEX (NAME(CSF.CSFTKDS.INDEX))
/*
```

You can change and use the Job Control Language according to the needs of your installation. For more information about allocating a VSAM data set, see *z/OS DFSMS Access Method Services for Catalogs*.

Steps to 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 previously (in Step 3 on page 15 and Step 4 on page 17).
 - 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 32.)
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   */
/*                                     */
/*   5694-A01                               */
/*                                     */
/*   (C) COPYRIGHT IBM CORP. 1990, 2007    */
/*                                     */
/*   STATUS = HCR7740                       */
/*                                     */
CKDSN(CSF.CSFCKDS)
PKDSN(CSF.CSFCKDS)
COMPAT(NO)
SSM(NO)
```

```

KEYAUTH(NO)
CKTAUTH(NO)
CHECKAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(64)

```

- CSFPRM01 (batch setup for SMP/E)

```

/*****
/*      LICENSED MATERIALS - PROPERTY OF IBM      */
/*      5694-A01                                  */
/*      (C) COPYRIGHT IBM CORP. 1990, 2003        */
/*      */                                          */
/*      THIS IS A SAMPLE OF THE ICSF OPTIONS DATASET */
/*      */                                          */
*****/
CKDSN(CSF.CSFCKDS)
PKDSN(CSF.CSFCKDS)
COMPAT(NO)
SSM(YES)
KEYAUTH(NO)
CKTAUTH(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 32 for descriptions of these parameters.

Use of system symbols in the options data set is 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 32 for additional information.

This 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='CSFCKDS')

```

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)
CKTAUTH(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 previous 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.

Steps to create the ICSF Startup Procedure

ICSF provides these 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=0M,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=0M,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. This 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 Cryptographic Services 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, 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 that:
 - 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.

Steps to 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 these steps:

1. Under the % OPTION ==> _ZCMD line, add this 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 this 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 "ATLIB DEACTIVATE APPLICATION(CLIST)"

"LIBDEF ISPSLIB"
"LIBDEF ISPLLIB"
"LIBDEF ISPLMLIB"
"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 topics in this publication, and *z/OS Cryptographic Services ICSF Administrator's Guide* for details about the remaining steps.

Steps to start ICSF for the first time

When starting ICSF for the first time, ensure that you have completed these steps. For additional information on starting ICSF for the first time, see Appendix C, "Helpful Hints for ICSF First Time Startup," on page 221.

- 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
- Specified the PKDS name in the installation options data set
- If PKCS #11 support is desired, create the TKDS
- Created a startup procedure
- Installed ICSF

Steps for initializing ICSF

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

1. Enter the START command and the startup procedure name. In this 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 "Steps to create the ICSF Startup Procedure" on page 23). See "Starting and stopping ICSF" on page 58 for more information about starting and stopping ICSF.

Note: CCF Systems Only: 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 Cryptographic Services 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 Cryptographic Services ICSF Administrator's Guide*.

3. When you start ICSF for the first time, you will see different messages depending on your system hardware.

CSFM122I

Message CSFM122I appears if you are running FMID HCR7750. It also appears if you are running FMID HCR7740 or earlier with APAR OA19007.

- **z9 EC, z9 BC and System z10 Enterprise Class:**

- First time startup messages before master keys have been loaded and the CKDS and PKDS have not been initialized:

- **z9 EC, z9 BC and System z10 Enterprise Class with CEX2C, without CEX2A**

S CSF

CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.

CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn.

CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.

CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.

CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION

CSFM001I ICSF INITIALIZATION COMPLETE

CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

Message CSFM434E will be issued for each CEX2C online.

- **z9 EC, z9 BC and System z10 Enterprise Class with CEX2C and CEX2A**

CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE

In addition to the messages issued for CEX2C, message CSFM435I will be issued for each active CEX2A if running HCR7730.

Note: Message CSFM508I will not be issued in this environment.

- First time startup messages before master keys have been loaded and sharing an initialized CKDS and PKDS:

- **z9 EC, z9 BC and System z10 Enterprise Class with CEX2C, without CEX2A**

S CSF

CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn.

CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.

CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION

CSFM001I ICSF INITIALIZATION COMPLETE

CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

Message CSFM434E will be issued for each CEX2C online.

- **z9 EC, z9 BC and System z10 Enterprise Class , with CEX2C and CEX2A**

CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE

In addition to the messages issued for the CEX2C, message CSFM435I will be issued for each active CEX2A if running HCR7730.

Note: CSFM508I will not be issued in this environment.

- Normal ICSF restart messages. Master key registers are valid and match the CKDS/PKDS.

- **z9 EC, z9 BC and System z10 Enterprise Class , with CEX2C, without CEX2A**

S CSF
 CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn,
 ACTIVE.
 CSFM431I BOTH MASTER KEYS CORRECT ON CRYPTO EXPRESS2 COPROCESSOR Epp,
 SERIAL NUMBER nnnnnnnn.
 CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.
 CSFM001I ICSF INITIALIZATION COMPLETE
 CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

Messages CSFM441I and CSFM431I will be issued for each active CEX2C.

- ***z9 EC, z9 BC and System z10 Enterprise Class with CEX2C and CEX2A***

CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE

In addition to the messages issued for the CEX2C, message CSFM435I will be issued for each active CEX2A.

Note: Message CSFM508I will not be issued in this environment.

- ***z9 EC, z9 BC and System z10 Enterprise Class with CPACF only (no CEX2C or CEX2A)***

S CSF
 CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.
 CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.
 CSFM507I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC COPROCESSORS ONLINE.
 CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.
 CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION
 CSFM001I ICSF INITIALIZATION COMPLETE
 CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

- ***z9 EC, z9 BC and System z10 Enterprise Class with CPACF and CEX2A***

S CSF
 CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE
 CSFM507I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC COPROCESSORS ONLINE.
 CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION
 CSFM001I ICSF INITIALIZATION COMPLETE
 CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

Message CSFM435I will be issued for each active CEX2A.

• **On a z990, or z890:**

- First time startup messages before master keys have been loaded and the CKDS and PKDS have not been initialized:

- ***z990, or z890 with a PCIXCC/CEX2C, without a PCICA***

S CSF
 \$HASP100 CSF ON STCINRDR
 IEF695I START CSF WITH JOBNAM CSF IS ASSIGNED TO USER
 \$HASP373 CSF STARTED
 IEF403I CSF - STARTED - TIME=15.34.03
 CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.
 CSFM419E INCORRECT MASTER KEY (SYM) ON PCI X CRYPTOGRAPHIC COPROCESSOR
 Xnn, SERIAL NUMBER nnnnnnnn.
 CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2 COPROCESSOR Epp,
 SERIAL NUMBER nnnnnnnn.
 CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.
 CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.
 CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION
 CSFM001I ICSF INITIALIZATION COMPLETE
 CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

Message CSFM419E will be issued for each PCIXCC/CEX2C online.
 The CEX2C is tolerated as a PCIXCC.

Message CSFM434E will be issued for each CEX2C online.

- **z990 or z890 with a PCIXCC/CEX2C, with a PCICA**

CSFM411I PCI CRYPTOGRAPHIC ACCELERATOR Ann IS ACTIVE

In addition to the messages issued for the PCIXCC/CEX2C, messages CSFM411I will be issued for each active PCICA.

Note: Message CSFM508I will not be issued in this environment.

- First time startup messages before master keys have been loaded and sharing an initialized CKDS and PKDS:

- **z990 or z890 with a PCIXCC/CEX2C, without a PCICA/CEX2A**

S CSF

\$HASP100 CSF ON STCINRDR

IEF695I START CSF WITH JOBNAM CSF IS ASSIGNED TO USER

\$HASP373 CSF STARTED

IEF403I CSF - STARTED - TIME=15.54.34

CSFM419E INCORRECT MASTER KEY (SYM) ON PCI X CRYPTOGRAPHIC COPROCESSOR Xnn, SERIAL NUMBER nnnnnnnn.

CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn.

CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.

CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION

CSFM001I ICSF INITIALIZATION COMPLETE

CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

Message CSFM419E will be issued for each PCIXCC/CEX2C online.
CEX2C is tolerated as a PCIXCC.

Message CSFM434E will be issued for each CEX2C online

- **z990 or z890 with a PCIXCC/CEX2C, with a PCICA/CEX2A**

CSFM411I PCI CRYPTOGRAPHIC ACCELERATOR Ann IS ACTIVE

In addition to the messages issued for the PCIXCC/CEX2C, message CSFM411I will be issued for each active PCICA.

Note: Message CSFM508I will not be issued in this environment.

- Normal ICSF restart messages. Master key registers are valid and match the CKDS/PKDS.

- **z990 or z890 with a PCIXCC/CEX2C, without a PCICA**

S CSF

\$HASP100 CSF ON STCINRDR

IEF695I START CSF WITH JOBNAM CSF IS ASSIGNED TO USER

\$HASP373 CSF STARTED

IEF403I CSF - STARTED - TIME=15.54.34

CSFM431I BOTH MASTER KEYS CORRECT ON CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn.

CSFM416I BOTH MASTER KEYS CORRECT ON PCI X CRYPTOGRAPHIC COPROCESSOR Xnn, SERIAL NUMBER nnnnnnnn.

CSFM440I PCI X CRYPTO COPROCESSOR Xpp, SERIAL NUMBER nnnnnnnn, ACTIVE.

CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn, ACTIVE.

CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.

CSFM001I ICSF INITIALIZATION COMPLETE

CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

Message CSFM416I will be issued for each PCIXCC online.

Message CSFM431I will be issued for each CEX2C online.

Message CSFM440I will be issued for each PCIXCC.

Message CSFM441I will be issued for each CEX2C.

- **z990 or z890 with a PCIXCC/CEX2C, with a PCICA**

CSFM411I PCI CRYPTOGRAPHIC ACCELERATOR Ann IS ACTIVE

In addition to the messages issued for the PCIXCC/CEX2C, message CSFM411I will be issued for each active PCICA.

Note: Message CSFM508I will not be issued in this environment.

On an z900 or z800:

Starting ICSF with a CCF (with or without a PCICC) for the first time, you will see these messages:

```
S CSF
$HASP100 CSF  ON STCINRDR
IEF695I START CSF  WITH JOBNAME CSF IS ASSIGNED TO USER
$HASP373 CSF STARTED
IEF403I CSF -  STARTED -  TIME=10.11.35
CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.
CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR C0, CPU 1 IS NOT VALID.
CSFM511E CRYPTOGRAPHY - MASTER KEY ON COPROCESSOR C1, CPU 4 IS NOT VALID.
CSFM106A CRYPTOGRAPHY - PKA MASTER KEYS ARE NOT VALID.
CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.
CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION
CSFM001I ICSF INITIALIZATION COMPLETE
```

You'll receive message CSFM511E for each Cryptographic Coprocessor Feature you have online.

Notes:

1. 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 previously (see step 3 on page 15) to use for the CKDS when starting ICSF.
2. 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.
3. 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.
4. You can add keys to the CKDS in several ways. See "The Cryptographic Key Data Set (CKDS)" on page 8 for details.
5. You can also write application programs to call services to perform cryptographic functions. See "Creating ICSF exits and generic services" on page 42 for details.

MK Initialization for SMP/E - CCF Systems Only

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 this example:

```
//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 these services are RACF protected in the CSFSERV class. If the services 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

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.

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. This 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 issues RACROUTE calls to perform the security access control checking and the results are logged in RACF SMF records that are cut by RACF. If you specify CHECKAUTH(NO), the authorization checks against resources in the CSFSERV class are not performed resulting in a significant performance enhancement for supervisor state and system key callers. However, the authorization checks are not logged in the RACF SMF records.

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.

CKTAUTH(YES or NO)

Decides if authentication will be performed for every CKDS record read from DASD.

YES Indicates authentication will be performed.

NO Indicates no authentication will be performed.

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 143 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)

Note: This keyword is no longer supported but is tolerated.

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).

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.

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.
- The PCIXCC/CEX2C has master key registers for the SYM-MK and ASYM-MK master keys. 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 *z/OS Cryptographic Services ICSF Overview*.

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:

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 3 on page 35 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 these 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. EXIT is responsible for logging in SMF the results of any authorization checks that are made.

Table 3. 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.
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.

Table 3. Exit Identifiers and Exit Invocations (continued)

Exit Identifiers	Exit Invocations
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.
CSF1GAV	Gets control during the PKCS11 get attribute value 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.
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.

Table 3. Exit Identifiers and Exit Invocations (continued)

Exit Identifiers	Exit Invocations
CSFPCI	Gets control during the PCI interface callable service.
CSFPCU	Gets control during the PIN Change/Unblock 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.
CSFRNGL	Gets control during the random number generate long callable service.
CSFRKD	Gets control during the retained key delete callable service.
CSFRKL	Gets control during the retained key list callable service.
CSFRKX	Gets control during the remote key export callable service.
CSF1SAV	Gets control during the PKCS11 set attribute value 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.
CSFTBC	Gets control during the trusted block create callable service.
CSFTCK	Gets control during the transform CDMF key callable service.
CSF1TRC	Gets control during the PKCS11 token record create callable service
CSF1TRD	Gets control during the PKCS11 token record delete callable service
CSF1TRL	Gets control during the PKCS11 token record list callable service
CSFTRV	Gets control during the transaction validation 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 5, “Installation Exits,” on page 87 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).

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: 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.

PKDSN(data-set-name)

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

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.

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 you specified REASONCODES(ICSF) and your service was processed on a PCICC or PCIXCC/CEX2C, a TSS reason code may be returned if there is no 1–1 corresponding ICSF reason code.

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

If you are running with the IBM @server zSeries 900, IBM @server zSeries 800, S/390 Enterprise Servers and S/390 Multiprise servers, you must perform these 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.

Note: The setting of the Environment Control Mask (ECM) enables SSM. Without TKE, the supplied ECM enables SSM. 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 *Support Element Operations Guide* and *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

SYSPLEXCKDS(YES or NO,FAIL(*fail-option*))

SYSPLEXCKDS(YES,FAIL(*fail-option*))

ICSF will join the ICSF sysplex group SYSICSF and this system will participate in sysplex-wide consistency for CKDS data.

SYSPLEXCKDS(YES,FAIL(YES))

Indicates ICSF initialization will end abnormally if the ICSF cross-system services environment cannot be established during ICSF initialization due to a failure creating the CKDS latch set or a failure to join the ICSF sysplex group. A value of either FAIL(YES) or FAIL(NO) will be ignored with SYSPLEXCKDS(NO,...).

SYSPLEXCKDS(YES,FAIL(NO))

Indicates ICSF initialization processing will continue even if the request to create a CKDS latch set fails or the request to join the ICSF sysplex group fails. The system will not be notified of updates to the CKDS by other members of the ICSF sysplex group.

SYSPLEXCKDS(NO,FAIL(*fail-option*))

CKDS update processing proceeds as it does today (i.e. no Cross-System Services task will be initialized, nor will any XCF signalling be performed when an update to a CKDS record occurs).

If you do not specify the SYSPLEXCKDS option, the default value is SYSPEXCKDS(NO,FAIL(NO)).

SYSPLEXTKDS(YES or NO,FAIL(*fail-option*))

Note: TKDSN needs to be specified for this to work. See on page 41.

SYSPLEXTKDS(NO,FAIL(*fail-option*))

Indicates no XCF signalling will be performed when an update to a TKDS record occurs.

SYSPLEXTKDS(YES,FAIL(*fail-option*))

Indicates the system will be notified of updates made to the TKDS by other members of the sysplex who have also specified SYSPLEXTKDS(YES,FAIL(*fail-option*)).

SYSPLEXTKDS(YES,FAIL(YES))

Indicates ICSF will terminate abnormally if there is a failure creating the TKDS latch set.

SYSPLEXTKDS(YES,FAIL(NO))

Indicates ICSF initialization processing will continue even if the request to create a TKDS latch set fails with an environment failure. This system will not be notified of updates to the TKDS by other members of the ICSF sysplex group.

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

TKDSN(data-set-name)

The name of an existing TKDS or an empty VSAM data set to be used as the TKDS. If the TKDSN option is not specified in the ICSF Installation Options Data Set no PKCS #11 services will be provided.

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 or PCI X Cryptographic Coprocessor/Crypto Express2 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.

The User Defined Extension (UDX) is responsible for logging in SMF the results of any authorization checks that were made.

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 (CICS 4.1 or higher) installed. It specifies 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 with CCFs/PCICCs and CSFWTL01 for systems with

PCIXCCs/CEX2Cs. The sample data set contains the same entries as the default ICSF CICS Wait List (i.e., 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 these conditions.

- Non-CICS customers will not specify a WAITLIST keyword.
- You must ensure, however, that if you have any existing CICS applications which invoke any of the ICSF services in the Wait List and if these applications were linked with ICSF stubs at a pre-OS/390 V2R10 level, then these applications must be re-linked with the current ICSF stubs.

If running on a z990, z890, z9 EC or z9 BC however, you must also ensure that any existing CICS applications which invoke any of these services are re-linked to ensure that the correct version of the stub is used: CSNBCKI, CSNBCKM, CSNBDEC, CSNBENC, CSNBKYTX, CSNBMGN, CSNBMVR, CSNBPEXX, CSNBRNG

- CICS customers who do not want to make use of CICS TRUE must either not enable the TRUE or must specify a WAITLIST keyword and point to an empty wait list data set (or 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 or CSFWTL01 of SYS1.SAMPLIB. The WAITLIST keyword in the Installation Options Data Set should be set to point to this modified data set. If you have any existing CICS applications which invoke any of the ICSF services in the Wait List and if these applications were linked with ICSF stubs at a pre-OS/390 V2R10 level, then these applications must be re-linked with the current ICSF stubs.

If running on a z990, z890, z9 EC or z9 BC any existing CICS applications which invoke any of these services are re-linked to ensure that the correct version of the stub is used: CSNBCKI, CSNBCKM, CSNBDEC, CSNBENC, CSNBKYTX, CSNBMGN, CSNBMVR, CSNBPEXX, CSNBRNG.

For additional information on the CICS Attachment Facility, see Appendix B, “CICS-ICSF Attachment Facility,” on page 215.

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.

Dispatching priority of ICSF

To avoid performance problems, the dispatching priority of ICSF should be set at least as high as that of the highest task using ICSF.

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 “service exits” on page 91 for more information.)

Chapter 3. Migration

This topic 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.

Attention: Although you are migrating to a new release, you should review the information in Chapter 2, “Installation, Initialization, and Customization,” on page 13; especially review customization steps that may have changed since your last migration.

An IPL is required when installing a new release of ICSF (it is possible for ICSF control blocks like the DACC and CCVT to persist in storage across an ICSF restart).

Consult these documents for information on migration and installation:

- *z/OS Migration*

This publication describes the migration tasks for z/OS at a system and element level.

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

- *z/OS Planning for Installation*

This publication 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 publication, 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 publication 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 topic describes some terms you may need to know as you use this publication.

Migration

Activities that relate to the installation of a new version or release of a program to replace a previous 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 these 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. These 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)

Migrating from earlier software releases

These topics describe common activities and considerations that should be considered when you migrate from:

- an earlier release of ICSF to HCR7750
- an earlier release of z/OS to z/OS V1R9

Callable Services

If you are running on a z990, z890, z9 EC, z9 BC and System z10 Enterprise Class server without a PCIXCC/CEX2C, see Appendix F. z890, z990, z9 EC, z9 BC and System z10 Enterprise Class without a PCIX Cryptographic Coprocessor or Crypto Express2 Coprocessor in the *z/OS Cryptographic Services ICSF System Programmer's Guide* for the callable services that are available to you.

Table 4. Summary of new and changed ICSF callable services

Callable service	Release	Description
One-Way Hash Generate (CSNBOWH/CSNBOWH1/CSNEOWH/CSNEOWH1)	HCR7750	Changed: Supports SHA-2 algorithms.
ICSF Query Facility (CSFIQF/CSFIQF6)	HCR7750	Changed: Supports SHA-2 algorithms.
PKA Decrypt (CSNFPKD/CSNFPKD)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
PKA Encrypt (CSNFPKE/CSNFPKE)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
PKA Key Extract (CSNFPKX/CSNFPKX)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
PKA Key Generate (CSNFPKG/CSNFPKG)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
PKA Key Import (CSNFPKI/CSNFPKI)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
PKA Key Token Build (CSNFPKB/CSNFPKB)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
PKA Key Token Change (CSNDKTC)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.

Table 4. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
PKDS Record Create (CSNDKRC/CSNFRKC)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class .
PKDS Record Write (CSNDKRW)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class .
PKDS Record Delete (CSNDKRD/CSNFRKD)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Remote Key Export (CSNDKRX)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Symmetric Key Export (CSNDSYX)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Symmetric Key Generate (CSNDSYG)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Symmetric Key Import (CSNDSYI)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Trusted Block Create (CSNDTBC)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Digital Signature Generate (CSNDDSG)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Digital Signature Verify (CSNDDSV)	HCR7750	Changed: Supports 4096-bit RSA keys on z9 EC, z9 BC and System z10 Enterprise Class.
Clear PIN Encrypt (CSNBCPE)	HCR7750	Changed: Supports ISO-3 PIN block format.
Encrypted PIN Generate (CSNBEPG)	HCR7750	Changed: Supports ISO-3 PIN block format.
Encrypted PIN Translate (CSNBPTR)	HCR7750	Changed: Supports ISO-3 PIN block format.
PIN Change/Unblock (CSNBPCU)	HCR7750	Changed: Supports ISO-3 PIN block format.
Secure Messaging for PINs (CSNBSPN)	HCR7750	Changed: Supports ISO-3 PIN block format.
Random Number Generate (CSNBRNG/CSNERNG/CSNBRNGL/CSNERNGL)	HCR7750	Changed: Permits the user to specify the length of the random number generated.
Token Record Create (CSFPTRC)	HCR7740	New: Support for PKCS #11.
Token Record Delete (CSFPTRD)	HCR7740	New: Support for PKCS #11.
Token Record List (CSFPTRL)	HCR7740	New: Support for PKCS #11.
Get Attribute Value (CSFPGAV)	HCR7740	New: Support for PKCS #11.
Set Attribute Value (CSFPSAV)	HCR7740	New: Support for PKCS #11.
Symmetric Key Decipher (CSNBSYD and CSNBSYD1)	HCR7740	Changed: New rule array keyword, CFB, is added to support Cipher Feedback Mode and PKCS-PAD, to support PKCS #7 padding for the AES and DES algorithms for cipher block chaining.

Table 4. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
Symmetric Key Encipher (CSNBSYE and CSNBSYE1)	HCR7740	Changed: New rule array keyword, CFB, is added to support Cipher Feedback Mode and PKCS-PAD, added to support PKCS #7 padding for the AES and DES algorithms for cipher block chaining.
MAC Generate (CSFNMGN and CSFNMGN1)	HCR7731	Changed: A new rule array keyword, TDES-MAC, is added to support ISO 16609 CBC mode T-DES. The padding used by TDES-MAC will be to pad X'00' to the next 8 byte multiple before passing the data to the cryptographic coprocessors.
MAC Verify (CSFNMVR and CSFNMVR1)	HCR7731	Changed: A new rule array keyword, TDES-MAC, is added to support ISO 16609 CBC mode T-DES. The padding used by TDES-MAC will be to pad X'00' to the next 8 byte multiple before passing the data to the cryptographic coprocessors.
Clear PIN Encrypt (CSNBCPE)	HCR7731	Changed: Support for PTR enhanced PIN security.
Clear PIN Generate Alternate (CSNBCPA)	HCR7731	Changed: Support for PTR enhanced PIN security.
Encrypted PIN Generate (CSNBEPG)	HCR7731	Changed: Support for PTR enhanced PIN security.
Encrypted PIN Translate (CSNBPTR)	HCR7731	Changed: Support for PTR enhanced PIN security and enhanced to produce return code 8,3016 - The value of the PAD data is not valid.
Encrypted PIN Verify (CSNBPVR)	HCR7731	Changed: Support for PTR enhanced PIN security.
PIN Change/Unblock (CSNBPCU)	HCR7731	Changed: Support for PTR enhanced PIN security.
Trusted Block Create (CSNDTBC)	HCR7731	New: Support for remote key load on z9 EC and z9 BC.
Remote Key Export (CSNDRKX)	HCR7731	New: Support for remote key load on z9 EC and z9 BC.
PKA Key Import (CSNFPKI)	HCR7731	Changed: Support for remote key load.
Digital Signature Verify (CSNDDSV)	HCR7731	Changed: Support for remote key load. A new rule array keyword, TPK-ONLY, allows a Trusted Block, with a Public Key Section, to be used to verify a signature.
PKA Key Token Change (CSNDKTC)	HCR7731	Changed: Support for remote key load.
PKA Key Generate - Permit Regeneration Data (CSNDPKG)	HCR7731	New: Added new control point in order to use regeneration data to create a particular RSA private-public key-pair.
PKA Key Generate - Permit Regeneration Data for Retained Keys (CSNDPKG)	HCR7731	New: Added new control point in order to use regeneration data to create a particular RSA private-public key-pair.
PKA Decrypt (CSNDPKD)	HCR7730	Changed: Support for CEX2A on z9 EC and z9 BC.
PKA Encrypt (CSNDPKE)	HCR7730	Changed: Support for CEX2A on z9 EC and z9 BC.
Digital Signature Verify (CSNDDSV)	HCR7730	Changed: Support for CEX2A on z9 EC and z9 BC.
Symmetric Key Encrypt (CSNBSYE)	HCR7730	Changed: Supports clear AES keys passed by CKDS label. Supports AES CPACF encryption for 128-bit keys.
Symmetric Key Encipher (CSNBSYE1)	HCR7730	Changed: Supports clear AES keys passed by CKDS label. Supports AES CPACF encryption for 128-bit keys.

Table 4. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
Symmetric Key Decrypt (CSNBSYD)	HCR7730	Changed: Supports clear AES keys passed by CKDS label. Supports AES CPACF encryption for 128-bit keys.
Symmetric Key Decipher (CSNBSYD1)	HCR7730	Changed: Supports clear AES keys passed by CKDS label. Supports AES CPACF encryption for 128-bit keys.
Key Token Build (CSNBKTB)	HCR7730	Changed: Supports CLRAES keys.
Key Record Write (CSNBKRW)	HCR7730	Changed: Supports CLRAES keys.
One Way Hash (CSNBOWH/CSNBOWH1)	HCR7730	Changed: Supports SHA-256 on the CPACF on z9 EC and z9 BC.
Key Record Read (CSNBKRR)	HCR7720	Changed: Support for enhanced key management for Crypto Assist instructions.
Key Record Write (CSNBKRW)	HCR7720	Changed: Support for enhanced key management for Crypto Assist instructions.
Key Token Build (CSNBKTB)	HCR7720	Changed: Support for enhanced key management for Crypto Assist instructions.
Symmetric Key Decipher (CSNBSYD/CSNBSYD1)	HCR7720	Changed: Support for enhanced key management for Crypto Assist instructions.
Symmetric Key Encipher (CSNBSYE/CSNBSYE1)	HCR7720	Changed: Support for enhanced key management for Crypto Assist instructions.
Clear Key Import (CSNECKI)	HCR7720	Changed: Supports invocation in AMODE(64).
Decipher (CSNEDEC)	HCR7720	Changed: Supports invocation in AMODE(64).
Digital Signature Generate (CSNFDSG)	HCR7720	Changed: Supports invocation in AMODE(64).
Digital Signature Verify (CSNFDSV)	HCR7720	Changed: Supports invocation in AMODE(64).
Encipher (CSNEENC)	HCR7720	Changed: Supports invocation in AMODE(64).
ICSF Query Facility (CSFIQF6)	HCR7720	Changed: Supports invocation in AMODE(64).
Key Generate (CSNEKGN)	HCR7720	Changed: Supports invocation in AMODE(64).
Multiple Clear Key Import (CSNECKM)	HCR7720	Changed: Supports invocation in AMODE(64).
One Way Hash (CSNEOWH)	HCR7720	Changed: Supports invocation in AMODE(64).
PKA Decrypt (CSNFPKD)	HCR7720	Changed: Supports invocation in AMODE(64).
PKA Encrypt (CSNFPKE)	HCR7720	Changed: Supports invocation in AMODE(64).
PKA Key Generate (CSNFPKG)	HCR7720	Changed: Supports invocation in AMODE(64).
PKA Key Import (CSNFPKI)	HCR7720	Changed: Supports invocation in AMODE(64).
PKA Key Token Build (CSNFPKB)	HCR7720	Changed: Supports invocation in AMODE(64).
PKA Public Key Extract (CSNFPKX)	HCR7720	Changed: Supports invocation in AMODE(64).
PKDS Record Create (CSNFKRC)	HCR7720	Changed: Supports invocation in AMODE(64).

Table 4. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
PKDS Record Delete (CSNFKRD)	HCR7720	Changed: Supports invocation in AMODE(64).
Random Number Generate (CSNERNG)	HCR7720	Changed: Supports invocation in AMODE(64).
Retained Key Delete (CSNFRKD)	HCR7720	Changed: Supports invocation in AMODE(64).
Retained Key List (CSNFRKL)	HCR7720	Changed: Supports invocation in AMODE(64).
Symmetric Key Decipher (CSNESYD)	HCR7720	Changed: Supports invocation in AMODE(64).
Symmetric Key Encipher (CSNESYE)	HCR7720	Changed: Supports invocation in AMODE(64).

ICSF Key Data Sets

CKDS

When sharing a CKDS with CCF systems and the IBM @server zSeries 990, IBM @server zSeries 890, IBM System z9 EC, IBM System z9 BC and System z10 Enterprise Class the CKDS must be created on a CCF system.

Once new key types are added to the CKDS, these considerations apply when sharing the CKDS with non-OS/390 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 PCICC, PCIXCC or CEX2C installed.
- when clear DES or AES keys are added to the CKDS, RACF-protect all clear DES and AES keys by label name on all systems sharing the CKDS.
- generation of a clear DES or AES key using KGUP requires a PCIXCC/CEX2C.

PKDS

The PKDS must be initialized. Support to INITIALIZE PKDS, REENCIPHER PKDS and ACTIVATE PKDS is available in the Master Key Management Panels and the CSFPUTIL utility. CSFPUTIL performs the same functions as the panels.

For information on managing and sharing the PKDS in a sysplex environment, see *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521.

If the PKDS is shared with down-level systems, install the toleration APAR on those systems to allow continued sharing of the PKDS. ICSF will be able to interact with both sizes of the PKDS. The toleration APAR number is OA21807.

Attention: Beginning with HCR7750, LRECL length in the PKDS has increased. You must allocate a new PKDS with the longer LRECL so that internal I/O routines can handle 4096-bit tokens. Failure to migrate the PKDS before starting HCR7750 results in the following error messages: CFC0286, CSFM406A and CSFM407A. See "Migrating to a larger PKDS" on page 18 for more information.

Beginning with HCR7731 and for HCR7720 and above with APAR OA15156, the following PKDS key management panels are available from the ICSF Utility panel:

- Generate an RSA key pair PKDS record

- Delete an existing PKDS record
- Export an existing public key to an X.509 certificate stored in an MVS physically sequential data set
- Import a public key from an X.509 certificate stored in an MVS physically sequential data set.

Notes:

1. If a trusted block token is put into the PKDS by an HCR7731 ICSF system, a prior release of ICSF will not be able to reencipher the PKDS. When the prior release of ICSF comes across a trusted block token, reencipher will be terminated and a message will be issued indicating that some tokens were reenciphered.
2. To reencipher you must be running on z9 EC, z9 BC and System z10 Enterprise Class with Remote Key Export (RKX) support.

For additional information on ME and CRT tokens, see diagnosis reference information in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

TKDS

Beginning with HCR7740, these TKDS key management panels are available from the ICSF Utility panel:

- Listing tokens
- Creating a token
- Deleting a token
- Listing the objects of a token
- Viewing the attributes of an object
- Updating the attributes of an object
- Deleting an object

Access authorization of the new callable services will be determined via SAF calls. No support will be provided for invocation of an installation security exit for these new services. The CSFSERV class will control access to the token management callable services.

Key Tokens

- On the z990, z890, z9 EC, z9 BC and System z10 Enterprise Class, all RSA internal tokens are usable on PCIXCC or CEX2C hardware if they are encrypted under the current ASYM-MK.
- Existing DES internal key tokens can be used on all cryptographic coprocessors: CCF, PCICC and PCIXCC/CEX2C.

Installation Options Data Set

- CKTAUTH - decides if authentication will be performed for every CKDS record read from DASD. With a large CKDS, CKTAUTH(YES) could cause an impact to performance.
- DOMAIN - this 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. On a z990 or z890, if a PCI Cryptographic Accelerator, PCIXCC or CEX2C is not available, or on a z9 EC, z9 BC and System z10 Enterprise Class if a CEX2A or CEX2C is not available, the DOMAIN parameter is not required. If a PCICA, PCIXCC, CEX2C, or CEX2A, 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.

- SYSPLEXCKDS - this parameter is optional. It specifies whether this system, if a member of a multi-system sysplex using a shared CKDS, should participate in XCF signalling in order to maintain consistency of the CKDS data across the sysplex.
- SYSPLEXTKDS - this parameter is optional. It specifies whether this system, if a member of a multi-system sysplex using a shared TKDS, should participate in XCF signalling in order to maintain consistency of the TKDS data across the sysplex.
- TKDSN - this parameter is the name of an existing TKDS or an empty VSAM data set to be used as the TKDS. If the TKDSN option is not specified in the ICSF Installation Options Data Set no PKCS #11 services will be provided. The TKDS must be a key-sequenced data set with spanned variable length records and must be allocated on a permanently resident volume.

Function Restrictions

Retained keys are RSA private keys that are stored in a cryptographic coprocessor instead of in the public key storage data set. This change does not affect retained keys that you are currently using, that is, keys that are stored on the cryptographic coprocessor. However, starting with ICSF HCR7750, the ICSF services do not allow you to store in a cryptographic coprocessor RSA keys intended for key management use. Your applications can continue to store in the cryptographic coprocessor RSA private keys intended for signature usage. The modulus length of these private keys is limited to 2048-bits. ICSF HCR7750 introduces 4096-bit modulus RSA keys support.

The 2048-bit RSA keys may have a public exponent, e , in the range of $1 < e < 2^{2048}$, and e must be odd. The RSA public key exponents for 2049-bit to 4096-bit RSA keys are restricted to the values 3 and 65537.

CICS Attachment Facility

If you have the CICS Attachment Facility (CICS 4.1 or higher) installed and you specify your own CICS wait list data set, you need to modify the wait list data set to include the new callable services.

With a PCI Cryptographic Coprocessor, modify and include:

- HCR7740: CSF1GKP, CSF1PKS, CSF1PKV, CSF1SAV, CSF1TRC, CSF1TRD, CSF1UWK, CSF1WPK
- HCR7731: CSFTBC, CSFRKX

Note: If no Wait List is specified on a z900 or z800, the default wait list will be used. (See sample CSFWTL00 for the contents of the default wait list for z900 or z800.)

With a PCI X Cryptographic Coprocessor or Crypto Express2 Coprocessor, modify and include:

- HCR7750: CSFRNGL
- HCR7740: CSF1GKP, CSF1GSK, CSF1PKS, CSF1PKV, CSF1SAV, CSF1TRC, CSF1TRD, CSF1UWK, CSF1WPK
- HCR7731: CSFTBC, CSFRKX

Note: If no Wait List is specified on a z990, z890, z9 EC, z9 BC and System z10 Enterprise Class the default wait list will be used. (See sample CSFWTL01 for the contents of the default wait list for z990, z890, z9 EC, z9 BC and System z10 Enterprise Class.)

Special Secure Mode

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

Note: On a CCF system, if a PCICC is available and the modulus bit length of the RSA public key is greater than or equal to 512 bits, then special secure mode is not required for CSNDSYG IM form.

Resource Manager Interface (RMF)

Support to enable RMF to provide performance measurements on these selected ICSF services and functions that use Direct Access Crypto (DAC) CCF instructions is available. On a z990, z890, z9 EC, z9 BC and System z10 Enterprise Class with a PCIXCC/CEX2C, the measurements refer to these services processing on PCIXCCs/CEX2Cs except for one-way hash. One-way hash is processed on CPACF.

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

TKE Workstation

The Trusted Key Entry (TKE) workstation (Version 4.0 or higher) is available on the IBM @server zSeries 990 IBM @server zSeries 890, z9 EC, z9 BC and System z10 Enterprise Class . It can also be used to provide key management on the IBM @server zSeries 900 and IBM @server zSeries 800.

TKE Version 5.0 is available for z9 EC, z9 BC, z990, z890, z900 and z800.

Operational key entry for the PCIXCC/CEX2C on the z990, z890, z9 EC, z9 BC and System z10 Enterprise Class was introduced with TKE V4.1.

Crypto Express2 support is provided beginning with TKE 4.2 with FMID HCR7720. TKE 4.0 and 4.1 will always recognize a CEX2C as a PCIXCC.

The TKE workstation (Version 3 and Version 4) use the IBM 4758 card. The TKE workstation Version 5 uses the IBM 4764 card. For detailed information, refer to *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

TKE Version 3.1 and 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. 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.

All of the mentioned 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 the Cryptographic Coprocessor Feature, the PCI Cryptographic Coprocessor, the PCI X Cryptographic Coprocessor or the Crypto Express2 Coprocessor. For functions which can be executed on the PCI Cryptographic Coprocessor or PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor, enablement of the function requires that the function be enabled through SAF and through the access control point in the DEFAULT Role. For additional details, see “TKE Version 4.x and Higher and Access to Callable Services.”

These are access control points for PCICCs.

TKE Version 4.x and Higher and Access to Callable Services

Access to services that are executed on the PCI X Cryptographic Coprocessor or Crypto Express2 Coprocessor is through Access Control Points in the DEFAULT Role. To execute callable services on the PCIXCC/CEX2C, access control points must be enabled for each service in the DEFAULT Role. 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 PCIXCC/CEX2C.

New TKE users and non-TKE users have all* access control points enabled. If you are migrating from TKE V4.0 or TKE V4.1 or TKE V4.2 to TKE V5.0 and have a PCIXCC/CEX2C, all your current access control points will remain the same and any new applicable access control points will not be enabled.

Note: *Access control points DKYGENKY-DALL and DSG ZERO-PAD unrestricted hash length and PTR enhanced PIN security are always disabled in the DEFAULT role for all customers (TKE and Non-TKE). A TKE Workstation is required to enable these access control points.

HCR7731

In order to use ATM Remote Key Loading, TKE users will have to enable the access control points for these functions:

- PTR Enhanced PIN Security - This access control point is available to previous releases of ICSF through APAR 0A13764.
- Trusted Block Create - API Keyword = Inactive
- Trusted Block Create - API Keyword = Active
- Public Key Import - Source Key Token = Trusted Block
- Public Key Import - Source Key Token = PKA96 Key Token

- Remote Key Export

TKE Enablement from the Support Element

On z890 or systems running with May 2004 or higher version of Licensed Internal Code or a z9 EC, z9 BC and System z10 Enterprise Class system running with MCL 029 Stream J12220 or higher version of Licensed Internal Code, you must enable TKE commands on each PCIXCC or CEX2C card from the support element. This is true for new TKE users and those upgrading to this level of LIC. See *Support Element Operations Guide*, SC28-6820 and *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*, SA23-2211 for more information.

Migrating from the IBM @server zSeries 800 and the IBM @server zSeries 900

Callable Services

These services are only available on the IBM @server zSeries 800 and the IBM @server zSeries 900. These services are not supported with a PCIXCC/CEX2C installed.

- ANSI X9.17 EDC Generate (CSNAEGN)
- ANSI X9.17 Key Export (CSNAKEX)
- ANSI X9.17 Key Import (CSNAKIM)
- ANSI X9.17 Key Translate (CSNAKTR)
- ANSI X9.17 Transport Key Partial Notarize (CSNAKTR)
- Ciphertext Translate (CSNBCTT)
- PKSC Interface Service (CSFPKSC)
- Transform CDMF Key (CSNBTCK)
- User Derived Key (CSFUDK)

Functions Not Supported

This topic lists functions not supported with a PCIXCC/CEX2C installed.

1. There is no KMMK (key management master key).
2. The Commercial Data Masking Facility (CDMF) is no longer supported. The CDMF keyword on KGUP control statements and panels is no longer supported.
3. The Public Key Algorithm Digital Signature Standard is not supported. This affects callable services CSNDPKG, CSNDPKI, CSNDDSG, and CSNDDSV.
4. The PBVC keyword is not supported on a PCIXCC/CEX2C. This affects callable services Clear PIN Generate Alternate (CSNBCPA), PIN Translate (CSNBPTR) and PIN Verify (CSNBPVR).

Setup Considerations

This topic lists setup changes that should be considered when installing a PCIXCC/CEX2C.

Consideration should be given to:

1. CICS wait list should be updated for services now executing on PCIXCCs/CEX2Cs. The sample CICS wait list, CSFWTL01, supplied by IBM includes these services and can be used as a reference.
2. PKDS initialization is required.
3. New options data set keyword CKTAUTH.

4. A CKDS initialized on a z990, z890, z9 EC, z9 BC and System z10 Enterprise Class cannot be used on CCF systems.
5. If sharing a PKDS with a PCICC and PCIXCC/CEX2C, delete the PKDS records for labelnames of retained keys on PCICCs no longer in use.
6. Customers who run CSFEUTIL to setup ICSF for automated electronic delivery process no longer need to execute CSFEUTIL on a z990, z890, z9 EC or z9 BC system. SHA-1 is available on z990, z890, z9 EC, z9 BC and System z10 Enterprise Class without entering ICSF master keys.
7. New options data set keyword SYSPLEXCKDS option.
8. New options data set keyword SYSPLEXTKDS option.

Programming Considerations

This topic lists programming changes that should be considered when installing a PCIXCC/CEX2C.

Consideration should be given to:

1. The DATAC key type should only be used with a PCIXCC/CEX2C on the IBM @server zSeries 990, IBM @server zSeries 890, z9 EC, z9 BC and System z10 Enterprise Class .
2. The PIN block format checking on PCIXCC/CEX2C is more rigorous than with a CCF.
For CSNBPVR, CSNBPTR and CSNBPA services, the input PIN block must have the correct format as specified in the PIN Profile parameter. On a CCF system, the PIN block format checking is incomplete.
For example, the REFORMAT processing mode of PIN Translate (CSNBPTR) may now fail on a PCIXCC/CEX2C when it was previously successful on a CCF. On a CCF, if input to the PIN verify service (CSNBPVR) is a malformed encrypted PIN block, the service will fail with return code 4, reason code 3028 (verification failed); on a PCIXCC/CEX2C, the service may fail with return code 8 and some appropriate reason code for invalid PIN format.
3. 512 to 2048 bit modulus for RSA keys is supported in all PKA services except SET services (Set Block Compose and Set Block Decompose).
4. All CCF functions are now executed on the PCIXCC/CEX2C. This may cause some impact on the performance of customer applications.
5. Reason codes from the PCIXCC/CEX2C may be different from previous cryptographic hardware.
6. With PCIXCCs/CEX2Cs, the requirement that caller must be in supervisor state to use NOCV tokens is lifted for the Key Record Write (CSNBKRW) service.
7. The z/OS SCHEDULE and IEAMSCHD macros are used to schedule SRBs. On the IBM @server zSeries 990, IBM @server zSeries 890, IBM System Enterprise Class or IBM System Business Class, since there are no CCFs on the system, applications should delete FEATURE=CRYPTO on the SCHEDULE and IEAMSCHD macros or the SRB being scheduled will not run.
8. External tokens that are export prohibited are imported differently on a z990, z890, z9 EC, z9 BC and System z10 Enterprise Class system with PCIXCCs/CEX2Cs. The imported internal token will have the same control vector as the external token with export prohibited. These tokens will only be usable on a z990, z890, z9 EC, z9 BC and System z10 Enterprise Class system with a PCIXCC/CEX2C or on CCF systems with PCICCs. On previous

hardware (CCF systems) the imported internal token had a control vector that allowed export, and export prohibition was enforced by the export flag in the token.

9. Prohibit Export service can now be used for MAC and MACVER keys.
10. New rule array keyword TDES-MAC added to the MAC Generate and MAC Verify services.
11. New rule array keywords, CFB and PKCS-PAD added to the Symmetric Key Decipher and Symmetric Key Encipher services.

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:

- **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 for the S/390 G5, G6, z800 and z900 servers and TKE Version 4 or higher for z800, z900, z890, z990, z9 EC, z9 BC and System z10 Enterprise Class servers. TKE Version 3, Version 4, and Version 5 supply 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.

- **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 these 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 an "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 PCICC, PCIXCC, or CEX2C. The administrator, through ICSF panels, performs UDX authorization processing on each PCI Cryptographic Coprocessor. Authorization is not LPAR specific. See *Managing User Defined Extensions in z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521 for additional information.

Support for writing your own UDX for a PCI Cryptographic Coprocessor is available. Development of a UDX for a PCIXCC or CEX2C requires a special contract with IBM. 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 the UDX parameter and Installation-Defined Callable Services in *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520 for additional details.

Chapter 4. Operating ICSF

Starting and stopping ICSF	58
Modifying ICSF	59
Using different configurations.	59
Configuring the IBM @server zSeries 990, IBM @server zSeries 890, IBM System z9 EC, IBM System z9 BC and System z10 Enterprise Class	59
Configuring the IBM @server zSeries 800 and the IBM @server zSeries 900	61
Single Image Mode	61
Logical Partition (LPAR) Mode	62
Adding and Removing Cryptographic Coprocessors	63
Adding Cryptographic Coprocessors	64
Steps for activating/deactivating cryptographic coprocessors	64
Steps to configure on/off cryptographic coprocessors	65
Steps for enabling/disabling cryptographic coprocessors (PCICC and PCIXCC/CEX2C)	65
Intrusion Latch on the PCICC and PCIXCC/CEX2C	66
Steps for enabling/disabling cryptographic coprocessors (CCF)	66
Performance considerations for using installation options	67
Dispatching priority of ICSF	68
VTAM session-level encryption	68
System SSL encryption	68
Access method services cryptographic option	69
Remote Key Loading.	69
Event Recording	70
System Management Facilities (SMF) Recording	70
ICSF Initialization (Subtype 1)	72
ICSF Status Change (Subtype 3)	72
Error Handling for Cryptographic Coprocessor Feature (Subtype 4)	73
Special Secure Mode Change (Subtype 5).	73
Master Key Part Entry (Subtype 6)	73
Operational Key Part Entry (Subtype 7)	73
CKDS Refresh (Subtype 8)	73
Dynamic CKDS Update (Subtype 9)	73
PKA Key Part Entry (Subtype 10)	74
Clear New Master Key Part Entry (Subtype 11)	74
PKSC Commands (Subtype 12)	74
Dynamic PKDS Update (Subtype 13).	74
Cryptographic Coprocessor Clear Master Key Entry (Subtype 14)	74
Cryptographic Coprocessor Retained Key Create or Delete (Subtype 15)	75
Cryptographic Coprocessor TKE Command Request or Reply (Subtype 16)	75
PCI Cryptographic Coprocessor Timing (Subtype 17)	75
Cryptographic Coprocessor Configuration (Subtype 18)	75
PCI X Cryptographic Coprocessor Timing (Subtype 19)	76
PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor Timing (Subtype 20)	76
ICSF Sysplex Group (Subtype 21).	76
Trusted Block Create (Subtype 22)	76
Token Data Set (TKDS) (Subtype 23)	77
Message Recording	77
Security Considerations.	77
Controlling the program environment	77
Controlling access to KGUP	78
Controlling access to the callable services	78

Controlling access to cryptographic keys	78
Scheduling changes for cryptographic keys	79
Controlling access to administrative panel functions	79
Controlling access to RACF SMF log records.	79
Debugging Aids	79
Component Trace	80
Examining the Trace Entry Buffer	80
ICSF System SVC 143	82
Abnormal Endings.	83
IPCS Formatting Routine	83

You use certain commands to operate ICSF. Also, there are different conditions for operating ICSF that you should consider. This topic describes the ICSF operating tasks.

Starting and stopping ICSF

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) and that the hash patterns in each coprocessor is the same as the hash pattern of the master key that enciphered the PKA key data set (PKDS).

CCF Systems

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, this 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.

PCIXCC/CEX2C Systems

Verification tests are also performed to ensure that the PCIXCC/CEX2C SYM-MK is the same on all PCIXCC/CEX2C coprocessors, and that the PCIXCC/CEX2C ASYM-MK is the same on all PCIXCC/CEX2C coprocessors.

If a SYM-MK master key does not match the CKDS, this occurs: ICSF starts and a message that indicates the verification failed for the indicated coprocessor appears on the console. The PCIXCCs/CEX2Cs will not be active.

If the ASYM-MKs do not match, or if they match but the hash pattern does not match the hash pattern in the PKDS, a message indicates that the PKA hash pattern in the PKDS does not match the system PKA hash pattern. PKA callable services are not enabled.

If the ASYM-MKs do match the hash pattern in the PKDS but the SYM-MK is not valid, then PKA callable services are not enabled. Once the SYM-MK becomes valid, the user will have to enable the PKA services or stop and restart ICSF.

Starting and Stopping ICSF

When ICSF successfully starts, a message that indicates that initialization is complete appears on the console.

This 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.

This 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 5, “Installation Exits,” on page 87 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 multiple cryptographic features of various types. This topic describes some of the different configurations available with the various servers.

Configuring the IBM @server zSeries 990, IBM @server zSeries 890, IBM System z9 EC, IBM System z9 BC and System z10 Enterprise Class

There is only LPAR mode on an IBM @server zSeries 990, IBM @server zSeries 890, z9 EC, z9 BC and System z10 Enterprise Class. 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, PCIXCCs/CEX2Cs, or CEX2As, or you plan to add PCICAs, PCIXCCs/CEX2Cs or CEX2As 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 PCICAs/PCIXCCs/CEX2Cs/CEX2As can be configured and shared across multiple partitions.

Note: The domain assigned to the TKE Host LPAR must be unique if TKE is to control all the coprocessor cards in the environment. No other LPAR can use the domain assigned to the TKE Host.

With the z990, z890, z9 EC or z9 BC, there is support for up to 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 and PCIXCCs or CEX2Cs. This is illustrated by LPAR 1 and LPAR 3 in Figure 2. They are both assigned to usage domain 0 but on two different PCICAs.

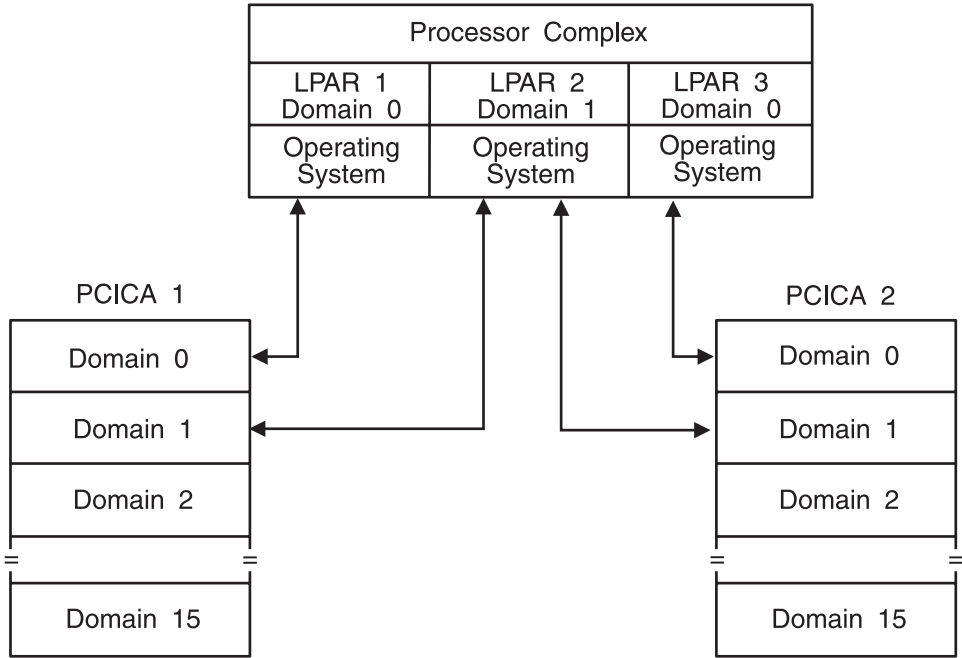


Figure 2. Two Crypto PCICAs on a Processor Complex Running in LPAR Mode

The example in Figure 2 shows that LPAR 2 has assigned access to Domain 1 on both PCICA 1 and PCICA 2. If you were to add another PCICA, PCIXCC or CEX2C to LPAR 2, Domain 1 on the new PCICA/PCIXCC/CEX2C would also be assigned.

A PCIXCC and PCICA configuration with domain sharing is illustrated by LPAR 1 and LPAR 3 in Figure 3.

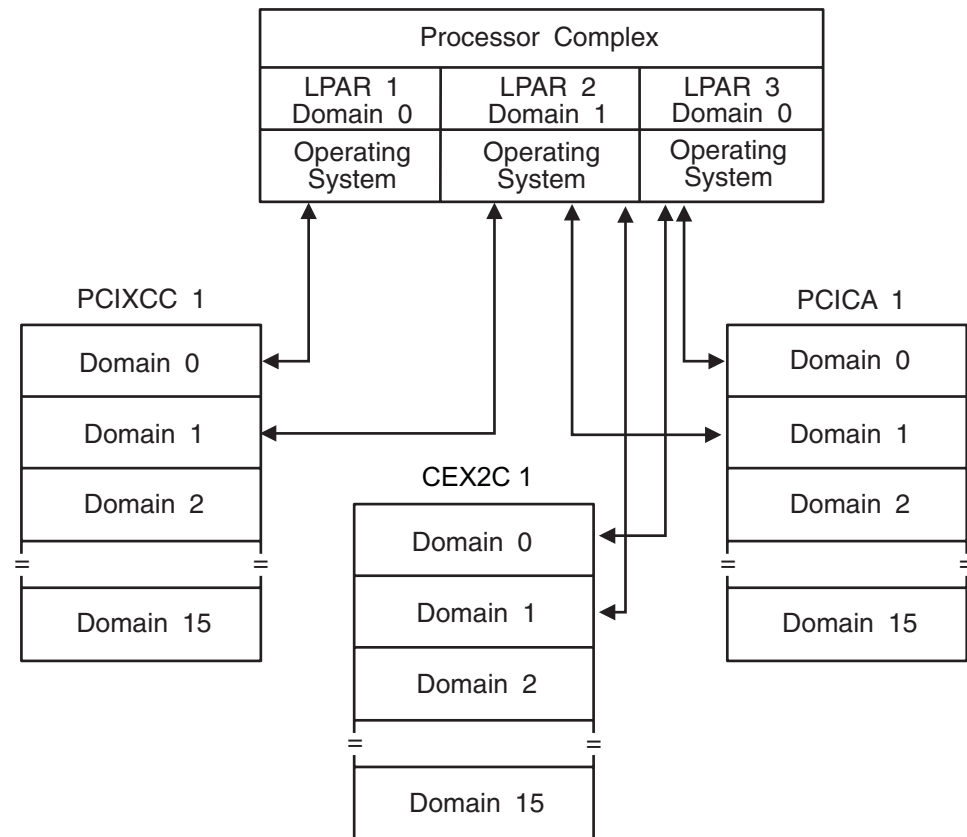


Figure 3. Multiple Crypto Coprocessors on a Complex Running in LPAR Mode

The example in Figure 3 shows that LPAR 2 has assigned access to Domain 1 on PCIXCC 1, CEX2C 1, and PCICA 1. LPAR 3 has assigned access to Domain 0 on CEX2C 1 and PCICA 1.

Configuring 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 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 these 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. 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 *z/OS Cryptographic Services ICSF System Programmer's Guide* for additional information on the DOMAIN parameter. The accessed domain on both coprocessors must have the same master key. Figure 4 on page 62 shows an

example single image mode configuration.

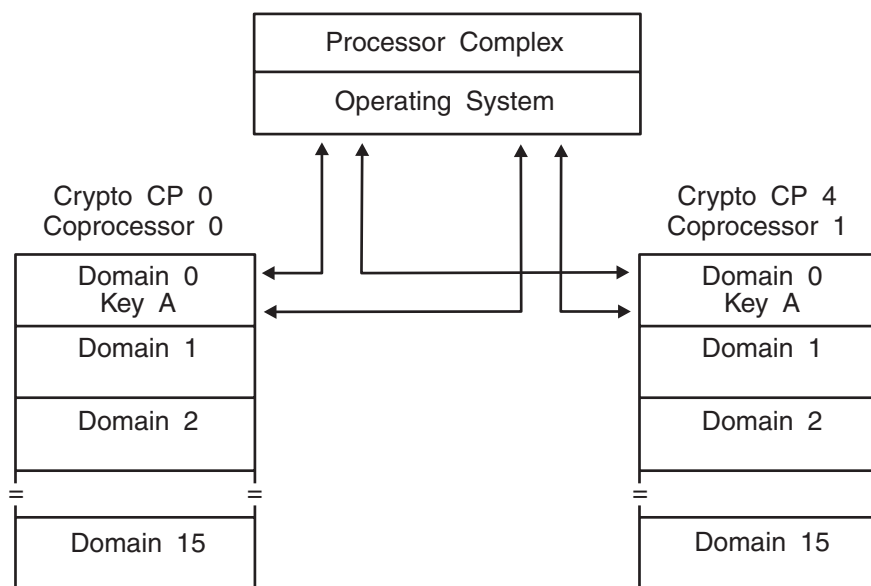


Figure 4. Two Crypto CPs on a Processor Complex Running in Single Image Mode

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. 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 *z/OS Cryptographic Services ICSF System Programmer's Guide* 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 5 on page 63 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.

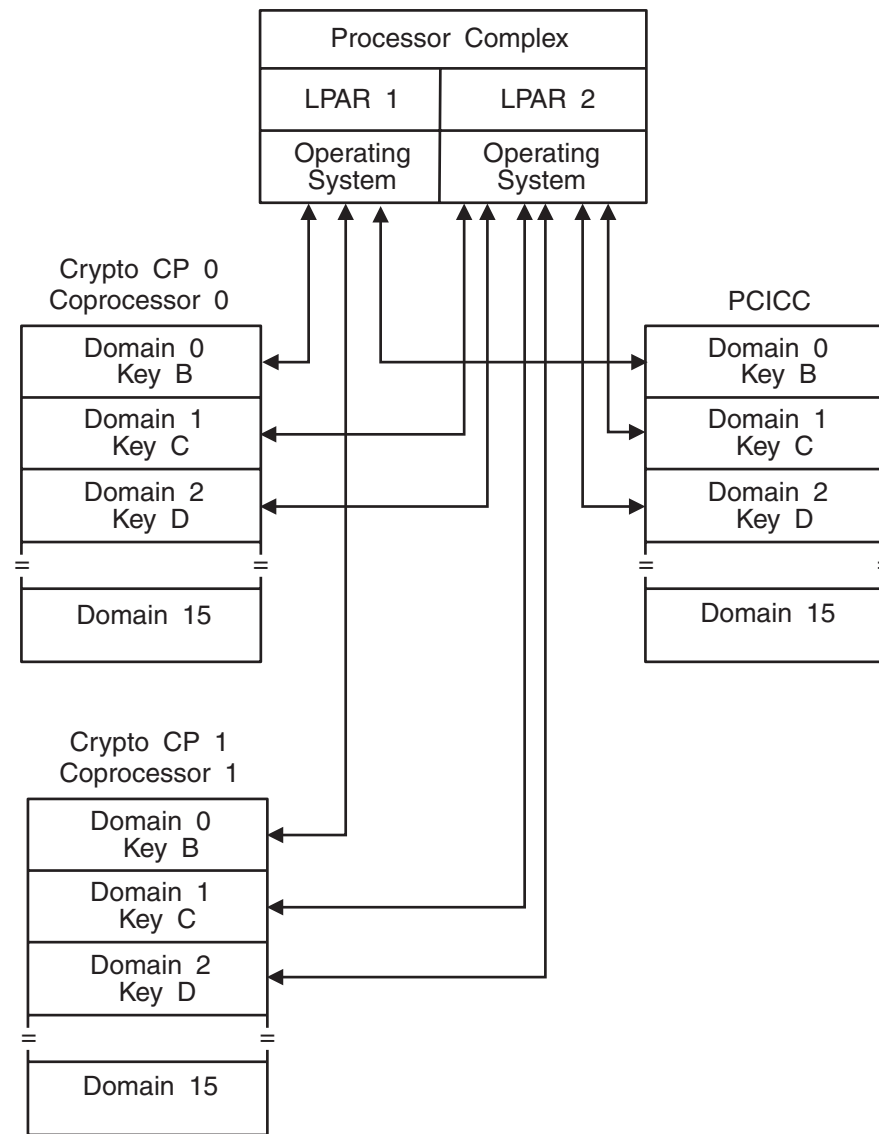


Figure 5. Two Crypto Coprocessors and one PCICC on a Processor Complex Running in LPAR Mode

Adding and Removing Cryptographic Coprocessors

It may become necessary for your installation to add or remove cryptographic coprocessors. This topic gives you a brief overview of the hardware implications. For more detailed information, refer to the *zSeries PR/SM Planning Guide* and the *zSeries Hardware Management Console Operations Guide (OS/2)*.

There are several terms associated with removing the cards. Use the Support Element (SE) panel to configure cryptographic coprocessors online and offline (standby). Use the ICSF Coprocessor Management panel from your TSO user ID to activate and deactivate cryptographic coprocessors. Use the TKE workstation to enable and disable cryptographic coprocessors.

Adding Cryptographic Coprocessors

You can dynamically add these cryptographic coprocessors: PCICA, PCICC, PCIXCC/CEX2C and CEX2A. If you are adding a PCIXCC/CEX2C, ensure that feature 3863 is installed on your IBM @server zSeries 990, IBM @server zSeries 890, z9 EC, z9 BC and System z10 Enterprise Class .

The cryptographic coprocessor number must be in the Candidates list of the LPAR Activation panel. **Configure On** the card. Each PCICC, PCIXCC/CEX2C, PCICA and CEX2A will display as ONLINE. Once the master keys are entered, they become ACTIVE. The PCICA and CEX2A will automatically become ACTIVE.

Steps for activating/deactivating cryptographic coprocessors

From your TSO userid, select option 1, Coprocessor Mgmt.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 1

Enter the number of the desired option.

  1  COPROCESSOR MGMT      - Management of Cryptographic Coprocessors
  2  MASTER KEY           - Master key set or change, CKDS/PKDS processing
  3  OPSTAT               - Installation options
  4  ADMINCNTL            - Administrative Control Functions
  5  UTILITY              - ICSF Utilities
  6  PPINIT               - Pass Phrase Master Key/CKDS Initialization
  7  TKE                  - TKE Master and Operational key processing
  8  KGUP                 - Key Generator Utility processes
  9  UDX MGMT             - Management of User Defined Extensions

      Licensed Materials - Property of IBM

      5694-A01 (C) Copyright IBM Corp. 1990, 2007. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure 6. Primary Panel

On the Coprocessor Management panel, you can select the coprocessors you want to activate or deactivate.

```
CSFGCMP0 ----- ICSF Coprocessor Management -----
COMMAND ==>

Select the coprocessors to be processed and press ENTER.
Action characters are: A, D, E, R, and S. See the help panel for details.

COPROCESSOR  SERIAL NUMBER                                STATUS
-----
- A06                                               ACTIVE
- A07                                               ACTIVE
- X02      42-K0001                                ONLINE
D X03      42-K0005                                ACTIVE
A X04      42-K0043                                DEACTIVATED
- X05      42-K0058                                DISABLED
- E08      93-X0604                                ACTIVE
- E09      93-X0608                                ONLINE
```

Figure 7. Coprocessor Management Panel

Note: This figure is for z890/z990. The coprocessors available for z9 EC and z9 BC, are Exx and Fxx.

When a PCICC, PCICA, PCIXCC/CEX2C or CEX2A is deactivated through the Coprocessor Management Panel, the card is only deactivated for that one LPAR.

Steps to configure on/off cryptographic coprocessors

To configure the PCICC, PCICA, PCIXCC/CEX2C and CEX2A cards online and offline, you must use the support element (SE) panel.

Before configuring a card offline, it is strongly recommended that you deactivate the card first from the ICSF Coprocessor Management panel. You need to 'deactivate' the card in ALL partitions that are using that card. This allow jobs to complete before the card is varied offline. You use the **Configure On/Off** service on the Support Element panel to take the card offline (standby).

After you configure the card offline from the SE panel, hit enter on the coprocessor management panel to verify that the card is offline. This configuring is done to remove and replace cards or to load new card code for the PCIXCC/CEX2C and PCICC cards.

To bring a card back online, use the SE panel again. If a card was deactivated and then brought offline (configured off), you will need to activate it again through the Coprocessor Management panel.

There are no z/OS operating commands to vary these devices.

Steps for enabling/disabling cryptographic coprocessors (PCICC and PCIXCC/CEX2C)

With TKE 3.0 or higher you can disable/enable the PCICCs. With TKE V4.0 or higher, you can disable/enable the PCIXCCs/CEX2Cs.

When a PCICA, PCICC, PCIXCC/CEX2C or CEX2A is deactivated through the Coprocessor Management Panel, the card is only deactivated for that one LPAR. When a PCICC, PCIXCC/CEX2C is disabled by TKE, the card is disabled for the entire system, not just the LPAR that issued the disable.

Intrusion Latch on the PCICC and PCIXCC/CEX2C

Under normal operation, the intrusion latch on a PCICC or PCIXCC/CEX2C is tripped when the card is removed. This causes all installation data, master keys, retained keys, roles and authorities to be zeroized in the card when it is reinstalled.

If a situation arises where a PCIXCC/CEX2C needs to be removed, for example, you need to remove your card for service, and you do not want the installation data to be cleared, perform this procedure to disable the PCIXCC/CEX2C before removing.

There is no similar procedure for the PCICC.

This process will require you to switch between the TKE application, the ICSF Coprocessor Management panel, and the Support Element.

1. Open an Emulator Session on the TKE workstation and logon to your TSO userid on the Host System where the PCIXCC/CEX2C will be removed.
2. From the ICSF Primary Option Menu on TSO, select Option 1 for Coprocessor Management.
3. Leave the Coprocessor Management panel displayed during the rest of this procedure. You will be required to hit ENTER on the Coprocessor Management panel at different times. DO NOT EXIT this panel.
4. Open the TKE Host where the PCIXCC/CEX2C will be removed. Open the PCIXCC/CEX2C. Click on Disable Crypto Module.
5. After the PCIXCC/CEX2C has been disabled from TKE, hit ENTER on the Coprocessor Management panel. The status should change to DISABLED.

Note: You do not need to deactivate a disabled card.

6. **Configure Off** the PCIXCC/CEX2C from the Support Element.
7. After the card has been taken Offline, hit ENTER on the Coprocessor Management panel. The status should change to OFFLINE.
8. Remove the PCIXCC/CEX2C. Perform whatever operation needs to be done. Replace the PCIXCC/CEX2C.
9. **Configure On** the PCIXCC/CEX2C from the Support Element.
10. When the initialization process is complete, hit ENTER on the Coprocessor Management panel. The status should change to DISABLED.
11. From the TKE Workstation Crypto Module General page, click on Enable Crypto Module.
12. After the PCIXCC/CEX2C has been enabled from TKE, hit ENTER on the Coprocessor Management panel. The Status should return to its original state. If the Status was ACTIVE in step 2, when the PCIXCC/CEX2C is enabled it should return to ACTIVE.

All installation data; master keys, retained keys, roles, and authorities should still be available. The PCIXCC/CEX2C data was not cleared with the card removal because it was Disabled first via the TKE workstation.

Steps for enabling/disabling cryptographic coprocessors (CCF)

With TKE V3.0 and higher, you can enable and disable the Cryptographic Coprocessor Feature.

Cryptographic functions can be disabled through the ECM Domains Controls 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: Status displayed on the coprocessor management panel will show DISABLED, not STANDBY.

Performance considerations for using installation options

You specify installation options in the installation options data set. Three installation options, CHECKAUTH, KEYAUTH, and CKTAUTH 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 can have a significant impact on performance when using the Crypto Express2 feature. 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.

The CKTAUTH option allows you to specify whether ICSF should authenticate an entry in the CKDS whenever ICSF reads the record from DASD. Customers with a large CKDS may experience a performance impact as each authentication requires a request to the PCIXCC/CEX2C. CKTAUTH has no effect on the KEYAUTH option.

The SYSPLEXCKDS option specifies whether this system, if a member of a sysplex using a shared CKDS, should participate in XCF signalling in order to maintain consistency of the CKDS data across the sysplex. If SYSPLEXCKDS(YES,FAIL(xxx)) is specified but the system is IPL'ed with PLEXCFG=LOCAL or PLEXCFG=MONOPLEX specified in IEASYSnn, (i.e., the system is not part of a multi-system sysplex), no XCF signalling will be invoked. No performance overhead will be incurred due to specifying SYSPLEXCKDS(YES,...) in a non-multi-system sysplex environment. In a multi-system sysplex environment, however, if SYSPLEXCKDS(YES,...) is specified to ensure consistency of CKDS data across the sysplex, the processing of sent/received XCF broadcast messages on each CKDS update operation (CKDS record create, write, delete) will have a performance impact.

The SYSPLEXTKDS option specifies whether sysplex-wide data consistency for the TKDS is desired. The SYSPLEXTKDS option is only in effect if the TKDSN option has also been specified.

The values of the option can be YES or NO, with the default being NO. If SYSPLEXTKDS(NO,FAIL(*fail-option*)) is specified, no XCF signalling will be performed when an update to a TKDS record occurs. If SYSPLEXTKDS(YES,FAIL(*fail-option*)) is specified, this system will be notified of updates made to the TKDS by other members of the sysplex who have also specified SYSPLEXTKDS(YES,FAIL(*fail-option*)).

The *fail-option* can be specified as either YES or NO. If FAIL(YES) is specified ICSF will terminate abnormally if there is a failure creating the TKDS latch set. If FAIL(NO) is specified ICSF initialization processing will continue even if the request to create a TKDS latch set fails with an environment failure. This system will not be notified of updates to the TKDS by other members of the ICSF sysplex group.

The TKDSN option is the name of an existing TKDS or an empty VSAM data set to be used as the TKDS. If the TKDSN option is not specified in the ICSF Installation Options Data Set, no PKCS #11 services will be provided. The TKDS must be a key-sequenced data set with spanned variable length records and must be allocated on a permanently resident volume.

Dispatching priority of ICSF

To avoid performance problems, the dispatching priority of ICSF should be set at least as high as that of the highest task using ICSF.

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.

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 greater than 16 megabytes.

System SSL encryption

ICSF supports System SSL encryption on all servers.

On the IBM @server zSeries 990, IBM @server zSeries 890, IBM System z9 EC, IBM System z9 BC and System z10 Enterprise Class, a PCICA or PCIXCC/CEX2C 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.

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 231 for more information in using this method.

Remote Key Loading

The process of remote key loading is loading DES keys to automated teller machines (ATMs) from a central administrative site. Because a new ATM has none of the bank's keys installed, getting the first key securely loaded is currently done manually by loading the first key-encrypting key (KEK) in multiple cleartext key parts. A new standard ANSI X9.24-2 defines the acceptable methods of doing this using public key cryptographic techniques, which will allow banks to load the initial KEKs without having to send anything to the ATMS. This method is quicker, more reliable and much less expensive.

Once an ATM is in operation, the bank can install new keys as needed by sending them enciphered under a KEK it installs at a previous time. Cryptographic architecture in the ATMs is not Common Cryptographic Architecture (CCA) and it is difficult to export CCA keys in a form understood by the ATM. Remote key loading will make it easier to export keys to non-CCA systems without compromising security.

In order to use ATM Remote Key Loading, TKE users will have to enable the access control points for these functions:

- Trusted Block Create - API Keyword = Inactive
- Trusted Block Create - API Keyword = Active
- Public Key Import - Source Key Token = Trusted Block
- Public Key Import - Source Key Token = PKA96 Key Token
- Remote Key Export

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 topic describes the events that ICSF records in the SMF record and describes where ICSF sends certain messages.

These records can be used with RACF SMF type 80 record to audit use of the callable services and the keys. The RACF type 80 records are extracted and formatted using the RACF SMF Unload Utility. See *z/OS Security Server RACF Auditor's Guide* for information on how to use this utility. For information about the formatted SMF records see *z/OS Security Server RACF Macros and Interfaces*.

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 these 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 part or key part register loaded using the TKE workstation
- TKE commands and responses are all audited through SMF 82 (TKE commands on the Cryptographic Coprocessor Feature use CSFPKSC. TKE commands on the PCICC or PCIXCC/CEX2C, 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 PCICC or PCIXCC/CEX2C
- You create or delete a retained key on a PCICC or PCIXCC/CEX2C
- The TKE workstation issues a PCICC or PCIXCC/CEX2C command request or receives a reply response from a PCICC or PCIXCC/CEX2C
- ICSF records processing times for PCICCs, PCIXCCs/CEX2Cs, PCICAs, and CEX2As
- A PCICA, PCICC, PCIXCC/CEX2C, or CEX2A is either brought online or taken offline
- ICSF issues IXCJOIN to join the ICSF sysplex group or issues IXCLEAVE to leave the sysplex group.
- The trusted block create callable service is used to create or activate a trusted block.

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 - 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. This is an example of CSFSMFJ.

```
//CSFSMFJ JOB <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=h1q.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=h1q.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=h1q.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,A,7,4,A),FORMAT=BI,SIZE=E4000
//*
//*-----*
//*   FORMAT TYPE 82 RECORDS                               *
//*-----*
//FMT      EXEC PGM=IKJEFT01,REGION=5128K,DYNAMNBR=100
//SYSPROC  DD DISP=SHR,DSN=h1q.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 this 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.

- CKDS record authentication (CKTAUTH) option
- User parameter (USERPARM) option
- PKDS name (PKDSN) option
- TKDS name (TKDSN) 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 this information:

- Processor number
- Coprocessor number
- Cryptographic domain number
- Master key version number

If a master key change or set occurs, subtype 3 also contains this 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 this 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 this information:

- The verification pattern for the master key part
- The verification pattern for the new master key
- 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.

Operational 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 this information:

- The ENC-ZERO verification pattern of the completed key for a PCIXCC/CEX2C or the CCF verification pattern
- A bit indicating whether the verification pattern is valid
- The cryptographic coprocessor domain number
- The cryptographic 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 this 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 this information:

- Name of the changed CKDS
- An indication of 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 this 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 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 this information:

- An indication of whether the hash patterns for the new master key register and new master key part are valid. (The new master keys register hash pattern is only valid once the final key part is entered.)
- An indication of whether the verification patterns for the new master key register and key part are valid. (The new master key verification pattern is valid only after 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 this 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 this information:

- The name of the changed PKDS
- An indication of the operation performed.
- The name of the changed entry in the PKDS

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 PCICC or PCIXCC/CEX2C. Subtype 14 contains this information:

- The master Key (SYM-MK or ASYM-MK; NMK or key part) valid indicator
- The indicator for a PCICC or PCIXCC/CEX2C
- The new master key verification pattern

- The key part verification pattern
- The cryptographic coprocessor processor number
- The cryptographic coprocessor serial number
- The domain index

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 PCICC or PCIXCC/CEX2C. Subtype 15 contains this information:

- The operation performed (created, deleted from PCI, deleted from PKDS)
- The indicator for a PCICC or PCIXCC/CEX2C
- The retained key label
- The cryptographic coprocessor processor number
- The cryptographic coprocessor serial number
- The domain index

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 PCICC or PCIXCC/CEX2C or receives a reply response from a PCICC or PCIXCC/CEX2C. Subtype 16 contains this information:

- The indicator for request or reply
- The indicator for a PCICC or PCIXCC/CEX2C
- The cryptographic coprocessor processor number
- The cryptographic coprocessor serial number
- The 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
- The length of the fixed audit data
- The number of relocate sections
- The function id
- The function return code
- The function description - describes the function id.

PCI Cryptographic Coprocessor Timing (Subtype 17)

ICSF periodically records processing times for PCI Cryptographic Coprocessor operations in subtype 17. Subtype 17 contains this 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 PCICA, PCICC, PCIXCC, CEX2C or CEX2A is brought online or taken offline. Subtype 18 contains this information:

- The operation performed (coprocessor brought online, taken offline, PCICA, PCICC, PCIXCC, CEX2C or CEX2A)
- The coprocessor number

- The PCICC , PCIXCC, CEX2C serial number or PCICA CEX2A number

PCI X Cryptographic Coprocessor Timing (Subtype 19)

ICSF periodically records processing times for PCIXCC operations in subtype 19. Subtype 19 contains this 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 PCIXCC, domain, and reference slot used by this operation
- The function code for this operation
- The PCIXCC processor number
- The PCIXCC serial number
- The PCIXCC domain
- A reference number that identifies an internal ICSF queue element

PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor Timing (Subtype 20)

ICSF periodically records processing times for PCIXCC, CEX2C or CEX2A operations in subtype 20. Subtype 20 contains this information:

- The device type
- 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 coprocessor, domain, and reference slot used by this operation
- The function code for this operation
- The PCIXCC, CEX2C or CEX2A processor number
- The PCIXCC or CEX2C serial number
- The PCIXCC, CEX2C or CEX2A domain
- A reference number that identifies an internal ICSF queue element

ICSF Sysplex Group (Subtype 21)

ICSF writes subtype 21 when ICSF joins or leaves the ICSF sysplex group. Subtype 21 contains this information:

- The name of the ICSF sysplex group
- The name of the sysplex member
- An indication of whether the member joined or left the sysplex group
- An indication of whether the join or leave was due to normal initialization/termination processing
- An indication of whether the leave was due to error recovery processing
- The time of the join or leave
- The name of the active CKDS

Trusted Block Create (Subtype 22)

ICSF writes subtype 22 when the Trusted Block Create callable services are invoked. Subtype 22 contains this information:

- Type of call, Active or Inactive
- If a Public Key Section was present in the Trusted Block Token
- ASID of the Caller
- If Input Trusted Block Token is in the PKDS, save it's Label

- If Output Trusted Block Token is in the PKDS, save it's Label
- If the Transport Key Token is in the CKDS, save it's Label

Token Data Set (TKDS) (Subtype 23)

ICSF writes subtype 23 when the Token Data Set (TKDS) record is updated (created, modified, deleted) of PKCS #11 tokens or token objects. Token Data Set callable services are invoked. Subtype 23 contains this information:

- The name of the changed TKDS
- An indication of the operation performed
- The name of the changed entry in the TKDS

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 Cryptographic Services 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 topic describes these 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 access to RACF SMF log records

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:

Security Considerations

```
RDEFINE PROGRAM * ADDMEM('CSF.SCSFMODE'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

Additional details on program control may be found in the "Program Control" topic 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 118.

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 Cryptographic Services ICSF Administrator's Guide*.

Access control points for specific functions may be enabled/disabled through the TKE workstation. See the *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide* 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 Cryptographic Services ICSF Administrator's Guide*.

When clear DES keys are added to the CKDS, RACF-protect all clear DES keys by label name on all systems sharing the CKDS.

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 118.

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 Cryptographic Services 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, these functions are also protected:

- Clear key entry (access can also be controlled through the TKE workstation, domain controls)
- Pass phrase MK/KDS initialization
- Administrative 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 ‘Viewing and Changing System Status’ in the *z/OS Cryptographic Services ICSF Administrator's Guide*, SA22-7521 for more information.

Controlling access to RACF SMF log records

For information about how to get RACF SMF log records UAUDIT, AUDIT, and GLOBALAUDIT see *z/OS Security Server RACF Auditor's Guide*.

You can extract RACF audit records that can be correlated to the ICSF log records. For more information on activating this auditing see *z/OS Security Server RACF Auditor's Guide*.

Debugging Aids

This topic contains information you can use when diagnosing problems on ICSF. This topic 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: These 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 these types of trace entries are always written to the buffer:

MISC:	Miscellaneous ICSF internal call
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 Misc Trace Entries: These items are traced for MISC entries:

Module:	Name of the ICSF module which issued this entry
X'8':	Returned PSMID
X'4A':	CCPA pointer
X'4B':	A 4 byte field.
ID:	Internal ICSF identifier

These items appear in both IPCS SUMMARY and IPCS FULL output.

Type-Specific Data for Service Trace Entries: These 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 these 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 this 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 this TRACE OFF command:

```
TRACE CT,OFF,COMP=CSF
```

Type-Specific Data for Instruction Trace Entries: These 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).

Sysplex CKDS Entry Types: You can activate these trace entries, in addition to the service and exit trace entry information that is always provided.

XCFMSGs: Traces the Send of an XCF message relating to an update to a CKDS in a sysplex
XCFMSGR: Traces the Receipt of an XCF message relating to an update to a CKDS in a sysplex
XCFENQX: Traces the return of control to the CKDS I/O subtask following the request for an exclusive ENQ on the SYSZCKDS.ckdsdsn resource. (The difference in the timestamp values between the XCFENQX entry and the XCFMSGs entry can be used to determine the delay due to multi-system effects of using the SYSPLEXCKDS option for sysplex-wide consistency of CKDS data.)

To activate the ICSF component trace for Sysplex CKDS trace entries, use this TRACE ON command:

```
TRACE CT,ON,COMP=CSF
```

Follow the TRACE ON command with this reply:

```
R nn,END
```

To deactivate Sysplex CKDS trace entries, use this TRACE OFF command:

```
TRACE CT,OFF,COMP=CSF
```

Security Considerations

Type-Specific Data for Sysplex CKDS Trace Entries: These items are traced for XCFMSGGS, XCFMSGGR, and XCFENQX entries:

ASCB:	ASCB address
TCB:	TCB address
ASID:	Address Space ID
GPRnn:	General Registers 0-12
GPRLLEN:	The length of the General Registers (8 for AMODE(64) operation, 4 otherwise)
CSS:	Address of ICSF Cross-System Services (CSS) block
SYSID:	The System ID of the system originating the XCF message
MsgInfo:	The message type and CKDS action (create, update, delete)
IOStatus:	The status of the CKDS I/O operation

These items appear in both IPCS SUMMARY and IPCS FULL output.

Sysplex TKDS Entry Types: You can activate these trace entries, in addition to the service and exit trace entry information that is always provided.

XCFTMSGGS:	Traces the broadcast of an XCF message related to TKDS I/O.
XCFTMSGGR:	Traces the Receipt of an XCF message relating to an update to a TKDS I/O.
XCFTENQ:	Traces the return of control to the TKDS I/O subtask following the request for an exclusive ENQ on the SYSZTKDS.TKDSdsn resource.

To activate the ICSF component trace for Sysplex TKDS trace entries, use this TRACE ON command:

```
TRACE CT,ON,COMP=CSF
```

Follow the TRACE ON command with this reply:

```
R nn,END
```

To deactivate Sysplex TKDS trace entries, use this TRACE OFF command:

```
TRACE CT,OFF,COMP=CSF
```

Type-Specific Data for Sysplex TKDS Trace Entries: These items are traced for XCFTMSGGS, XCFTMSGGR, and XCFTENQ entries:

ASCB:	ASCB address
TCB:	TCB address
ASID:	Address Space ID
GPRnn:	General Registers 0-12
GPRLLEN:	The length of the General Registers (8 for AMODE(64) operation, 4 otherwise)
CSS:	Address of ICSF Cross-System Services (CSS) block

These items appear in both IPCS SUMMARY and IPCS FULL output.

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 143 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 these cases only:

- 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

There is a new CTrace filter exit for ICSF. You can now issue these IPCS commands:

```
CTRACE COMP(CSF) OPTIONS((COUNTS,FAILURES))
CTRACE COMP(CSF) OPTIONS((COUNTS))
CTRACE COMP(CSF) OPTIONS((FAILURES))
```

COUNTS

Produces a list of services called and how often they were called.

FAILURES

Produces output for each failed ICSF service trace entry.

There is a new formatter for ICSF called CSFDATA. It is an IPCS VERBEXIT. To run it, enter:

```
VERBX CSFDATA 'options'
```

The supported options are:

- CELL
- CCPV
- CCPP
- CCPA
- CCPS
- CACB
- CCPD

If no options are specified you get VERBX CSFDATA Output:

No valid options were specified on VERBX CSFDATA.
Valid options are CELL,CCPV,CCPP,CCPA,CCPS,CACB,CCPD

Security Considerations

Sample output:

```
COMPONENT TRACE FULL FORMAT
COMP(CSF)
OPTIONS((FAILURES))
**** 08/15/2006
```

SYSNAME	MNEMONIC	ENTRY ID	TIME STAMP	DESCRIPTION
SYSAK	ASERVICE	00000006	12:58:13.679197	After call to service
	ASCB@....	00FCD980	TCB@.....	007BCA38 AS_id.... 0022
	Module...	CSFNENC	RCode....	00000008 Reason... 0000002F
	User.....	00000000	00000000	00000000 00000000 00000000
		00000000	00000000	00000000 00000000 00000000
		0000		
SYSAK	ASERVICE	00000006	13:07:02.162950	After call to service
	ASCB@....	00FCD980	TCB@.....	007BCA38 AS_id.... 0022
	Module...	CSFNYSI	RCode....	00000008 Reason... 00000042
	User.....	00000000	00000000	00000000 00000000 00000000
		00000000	00000000	00000000 00000000 00000000
		0000		

```
=====
COMPONENT TRACE FULL FORMAT
COMP(CSF)
OPTIONS((COUNTS))
```

```
ICSF COUNTS FROM CTRACE:
SERVICE CALLS_FOUND = 00000070
FAILING SERVICES     = 00000035
SERVICE  #SUCCESS  #FAILED
CSFSSTSW  00000001  00000000
CSFNPCV   00000001  00000000
CSFNPCM   00000001  00000000
CSFSTRL   00000002  00000000
CSFNENC   00000003  00000025
CSFNDEC   00000017  00000000
CSFNYSI   00000003  00000004
CSFNPKD   00000007  00000006
```

```
=====
COMPONENT TRACE FULL FORMAT
COMP(CSF)
OPTIONS((COUNTS,FAILURES))
**** 08/15/2006
```

SYSNAME	MNEMONIC	ENTRY ID	TIME STAMP	DESCRIPTION
SYSAK	ASERVICE	00000006	12:58:13.679197	After call to service
	ASCB@....	00FCD980	TCB@.....	007BCA38 AS_id.... 0022
	Module...	CSFNENC	RCode....	00000008 Reason... 0000002F
	User.....	00000000	00000000	00000000 00000000 00000000
		00000000	00000000	00000000 00000000 00000000
		0000		
SYSAK	ASERVICE	00000006	13:11:04.023615	After call to service
	ASCB@....	00FCD980	TCB@.....	007BCA38 AS_id.... 0022

```

Module... CSFNPKD   RCode.... 00000008 Reason... 00000041
User..... 00000000 00000000 00000000 00000000 00000000 00000000
          00000000 00000000 00000000 00000000 00000000 00000000
          0000

```

```

ICSF COUNTS FROM CTRACE:
SERVICE CALLS FOUND = 00000010
FAILING SERVICES    = 00000002
SERVICE  #SUCCESS  #FAILED
CSFNENC   00000003  00000001
CSFNPKD   00000007  00000001

```

```
=====
```

Sample output of ?CSFTRCE TYPE(MISC)

```

COMPONENT TRACE FULL FORMAT
COMP(CSF)
**** 08/15/2006

```

SYSNAME	MNEMONIC	ENTRY ID	TIME STAMP	DESCRIPTION
SYSAK	MISC	00000013	15:20:29.471700	Miscellaneous
	ASCB@....	00F58100	TCB@.....	007D4A70 AS_id.... 001E
	Module...	CSFACCPD	HEX8.....	00297D54 F800804D
	Hex4A....	7F44B3B0	Hex4B....	00E7F1F0 ID..... X10
SYSAK	MISC	00000013	15:20:29.471702	Miscellaneous
	ASCB@....	00F58100	TCB@.....	007D4A70 AS_id.... 001E
	Module...	CSFACCPD	HEX8.....	00297D54 F800804D
	Hex4A....	7F44B468	Hex4B....	ACD7F0F1 ID..... P01
SYSAK	MISC	00000013	15:20:29.471731	Miscellaneous
	ASCB@....	00FB1780	TCB@.....	007D54F8 AS_id.... 0029
	Module...	CSFGCCPN	HEX8.....	00297D54 F800804D
	Hex4A....	7F44B468	Hex4B....	B8D7F0F8 ID..... P08
SYSAK	MISC	00000013	15:20:29.476769	Miscellaneous
	ASCB@....	00FB0D00	TCB@.....	007D9138 AS_id.... 0030
	Module...	CSFASEND	HEX8.....	00307D91 3800804F
	Hex4A....	7F44B3B0	Hex4B....	00E7F1F1 ID..... X11
SYSAK	MISC	00000013	15:20:29.476769	Miscellaneous
	ASCB@....	00FB0D00	TCB@.....	007D9138 AS_id.... 0030
	Module...	CSFASEND	HEX8.....	00307D91 3800804F
	Hex4A....	7F44B468	Hex4B....	A4D7F1F0 ID..... P10

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 5 lists the control blocks you can display, the symbol IPCS recognizes for each control block, and a reference for the control block format.

Table 5. 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 10 on page 97.

Security Considerations

Table 5. *IPCS Symbols and Format References for the ICSF Control Blocks (continued)*

Control Block	Symbol	Format Reference
Cryptographic Communication Vector Table	CSFCCVT	See Table 59 on page 201.
Cryptographic Communication Vector Table Extension	CSFCCVE	See Table 60 on page 207.
Secondary Parameter Block	CSFASPB	See Table 14 on page 107.

For example, to format and display the ICSF Exit Name table issue this 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 this 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*.

Chapter 5. Installation Exits

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 topic 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 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 service, and when accessing a CKDS entry
- An exit called at various points during KGUP processing

These topics 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 these 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 92.

Exits for the services

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

You can use a service exit to change, augment, or replace processing or to bypass the IBM-supplied processing for the service entirely. “Services installation exits” on page 99 gives further details about exits for the services.

The PCF CKDS conversion program exit

The PCF conversion program changes a CKDS from PCF to ICSF CKDS format. See Chapter 7, “Migration from PCF to z/OS ICSF,” on page 143 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 112 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 115 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 services to manage keys on ICSF. A service can access a key in the in-storage CKDS by specifying a key label. For more information about the services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

When a 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 109 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 these points:

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

See “Security installation exits” on page 118 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 these 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 122 gives a more detailed description of the KGUP exit.

Entry and return specifications

All of the exits described in “Types of exits” on page 87 use standard linkage conventions on entry and return from the exits.

Registers at entry

The mainline exits have these 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 service exits have these 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 these 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 these 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 this 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.

service exits

ICSF calls the 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 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. This 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 these characteristics:

- Supervisor state
- Key 0
- APF-authorized
- TCB mode
- Address Space Control mode=access register mode
- AMODE(31) or AMODE(64)

The exit receives control in AMODE(64) if the service was invoked in AMODE(64); otherwise the exit receives control in AMODE(31). If you have a service exit for a service which supports invocation by an AMODE(64) caller, once HCR7720 is installed you should recode your exit to be sure it can handle being invoked in AMODE(64).

- 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 this 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 32. The EXIT keyword has this 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 8 illustrates the contents of register 0 and EXPB for the mainline exits.

Register 0

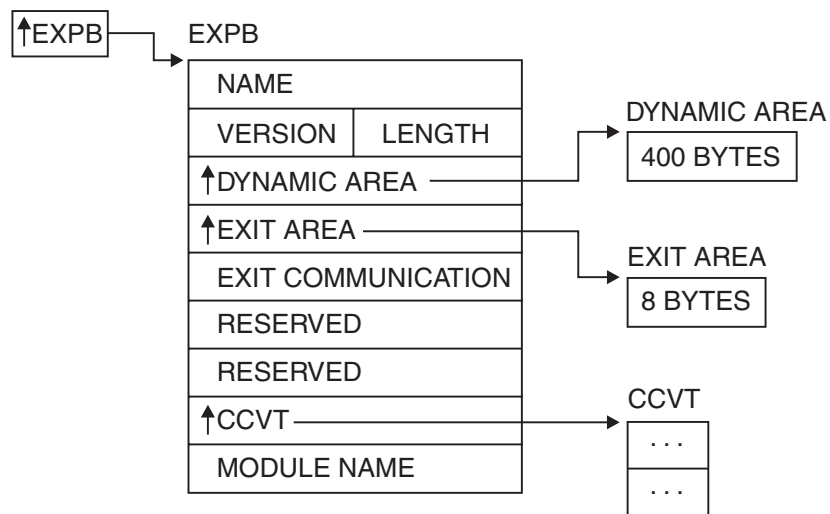


Figure 8. EXPB Control Block for Mainline Exits

Both the mainline exits and the services exits receive the address of EXPB in register 0. Some of the fields in EXPB are used only by the service exits and are reserved fields for the mainline exits.

The Exit Parameter Block

Table 6 describes the contents of the exit parameter block.

Table 6. 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.
4	2	Version. The version of the control block. This field contains the character string 01.

Table 6. EXPB Control Block Format for Mainline Exits (continued)

Offset (Dec)	Number of Bytes	Description
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 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 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 200 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 topics describe the parameters for each of the mainline exits.

CSFEXIT1: This table describes the parameters for CSFEXIT1:

Table 7. CSFEXIT1 Parameters

Parameter	Number of Bytes	Description
1	8	The data set name (DDNAME) of the installation options data set.

Table 7. CSFEXIT1 Parameters (continued)

Parameter	Number of Bytes	Description
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 8 describes these parameters.

Table 8. 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 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: <table><thead><tr><th>Bit</th><th>Meaning When Set On</th></tr></thead><tbody><tr><td>0</td><td>Special secure mode allowed.</td></tr><tr><td>1</td><td>Special secure mode enabled.</td></tr><tr><td>2</td><td>Reserved and must be zero.</td></tr><tr><td>3</td><td>Key authentication required.</td></tr><tr><td>4</td><td>The hardware has gone from active to inactive.</td></tr><tr><td>5</td><td>First start of ICSF during this IPL.</td></tr><tr><td>6</td><td>Security Server (RACF) checking required for authorized callers.</td></tr><tr><td>7</td><td>PCF coexistence.</td></tr></tbody></table> Bytes 2–4: Reserved	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.
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.																			
4	4	Address of the exit name table. Table 10 on page 97 describes the exit name table.																		

CSFEXIT4 and CSFEXIT5: Both CSFEXIT4 and CSFEXIT5 receive the same parameters. Table 9 on page 97 describes these parameters.

Table 9. 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 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: <table><thead><tr><th>Bit</th><th>Meaning When Set On</th></tr></thead><tbody><tr><td>0</td><td>Special secure mode allowed.</td></tr><tr><td>1</td><td>Special secure mode enabled.</td></tr><tr><td>2</td><td>Reserved and must be zero.</td></tr><tr><td>3</td><td>Key authentication required.</td></tr><tr><td>4</td><td>The hardware has gone from active to inactive.</td></tr><tr><td>5</td><td>First start of ICSF during this IPL.</td></tr><tr><td>6</td><td>Security Server (RACF) checking required for authorized callers.</td></tr><tr><td>7</td><td>PCF coexistence.</td></tr></tbody></table> Bytes 2–4: Reserved	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.
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.																			
4	4	Address of the exit name table. Table 10 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 10 describes the format of the exit name table.

Table 10. 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.
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.

Table 10. Format of the Exit Name Table (continued)

Offset (Dec)	Number of Bytes	Description																												
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: <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Exit has been requested by the installation.</td></tr><tr><td>1</td><td>Exit has been loaded.</td></tr><tr><td>2</td><td>Exit is active.</td></tr><tr><td>3</td><td>If exit fails, end ICSF.</td></tr><tr><td>4</td><td>If exit fails, do not call the exit again.</td></tr><tr><td>5</td><td>If exit fails, fail the service.</td></tr><tr><td>6</td><td>If exit fails, do nothing.</td></tr><tr><td>7</td><td>Exit has failed previously.</td></tr></table> Byte 2: <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>The exit should be called.</td></tr><tr><td>1</td><td>The exit is available to the installation.</td></tr><tr><td>2</td><td>If the security exit fails, fail the service.</td></tr><tr><td>3–7</td><td>Reserved.</td></tr></table>	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.	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.
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.																													
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.																												
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 these decimal return codes:

Return Code	Description
0	Proceed with initialization.
16	End ICSF.

CSFEXIT4 supports these decimal return codes:

Return Code	Description
0	Proceed with the STOP command.
4	Do not allow the STOP command to proceed.

CSFEXIT5 supports these decimal return codes:

Return Code	Description
0	Continue processing.
4	End ICSF.

Any return codes other than those listed cause ICSF to end abnormally.

Services installation exits

ICSF provides 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 services to customize processing. For a detailed description of the services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Use this general format to request a service:

```
CALL CSNBxxx (
    return_code
    ,reason_code
    ,exit_data_length
    ,exit_data
    ,parameter_5
    ,parameter_6
    .
    .
    .
    ,parameter_N)
```

Table 11 on page 101 lists the ICSF exit names for each of the services. The parameters that the application passes to a service are known as the service parameter list, and the parameters vary from service to service. “Parameters” on page 109 describes the services parameter lists in more detail.

Purpose and use of the exits

Each of the services has an installation exit. Each installation exit for a service has two exit points:

- **The Preprocessing exit point.** This exit point occurs after an application program calls a service, but before the 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 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 these 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 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 service by installing the load module that contains the exit into an APF-authorized library. ICSF uses this 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 32. The EXIT keyword has this syntax:

EXIT (ICSF name, load module name, FAIL (options))

The **ICSF name** portion of the keyword refers to the ICSF name for each service exit. Note that the ICSF name for each service exit is the same as its name. Table 11 lists the ICSF names for each of the service exits. Table 12 on page 103 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 this table, CSFPKSC (PKSC interface) and CSFPCI (PCI interface), are a part of the product-sensitive programming interface.

Table 11. Services and Their ICSF Names

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
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

Table 11. Services and Their ICSF Names (continued)

Service	ICSF Name
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
PIN change/unblock	CSFPCU
PKA Decrypt	CSFPKD
PKA Encrypt	CSFPKE
PKA Key Generate	CSFPKG
PKA Key Import	CSFPKI
PKA Key Token Change	CSFPKTC
PKA Public Key Extract	CSFPKX
PKCS11 Get Attribute Value	CSF1GAV
PKCS11 Set Attribute Value	CSF1SAV
PKCS11 Token Record Create	CSF1TRC

Table 11. Services and Their ICSF Names (continued)

Service	ICSF Name
PKCS11 Token Record Delete	CSF1TRD
PKCS11 Token Record List	CSF1TRL
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
Random Number Generate Long	CSFRNGL
Remote Key Export	CSFRKX
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
Transaction Validation	CSFTRV
Transform CDMF Key	CSFTCK
Trusted Block Create	CSFTBC
User Derived Key	CSFUDK
VISA CVV Service Generate	CSFCSG
VISA CVV Service Verify	CSFCSV

Notes:

1. The alias for the ANSI X9.17 key management services is CSNAxxx.
2. The aliases for the PKA services is CSNDxxx or CSNFxxx.
3. The aliases for the DES services is CSNBxxx or CSNExxx.

Table 12. 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 service gets the address of the exit parameter block (EXPB) in register 0. ICSF obtains and initializes an EXP for every service call. Figure 9 illustrates the contents of register 0, and Table 13 illustrates the EXPB for the service exits.

Register 1 contains the address of an address list. Each address in the list points to a parameter. "Parameters" on page 109 describes the service parameter list. The parameters the exit receives are the same parameters that are passed on the call to the service. For more information about the parameters for each service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Register 0

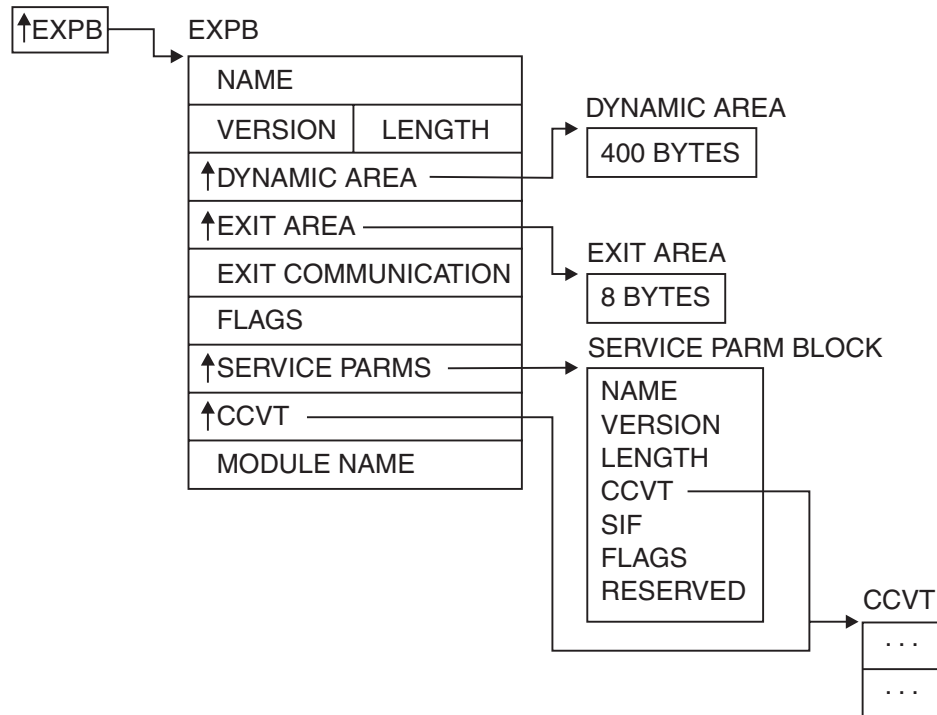


Figure 9. EXPB Control Block in the Service Exits

Exit parameter block

Table 13 describes the contents of the exit control block.

Table 13. EXPB Control Block Format for 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 13. EXPB Control Block Format for 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 service exit.

Table 13. EXPB Control Block Format for 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. <table><tr><th>Bit</th><th>Meaning When Set On/Off</th></tr><tr><td>0</td><td>Postprocessing invocation./Preprocessing invocation.</td></tr><tr><td>1</td><td>Reserved.</td></tr><tr><td>2</td><td>Use the return and reason code that the exit places in register 0 and register 15 as the service's return code/reason code. Do not use the exit's return code as the service return code in registers 0 and 15.</td></tr><tr><td></td><td>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 service's return and reason codes. See "Return Codes" on page 109 for more information about using exit return codes.</td></tr><tr><td>3</td><td>Do not call the postprocessing invocation of the service exit./Call the postprocessing invocation of the service exit.</td></tr><tr><td>4</td><td>Bypass the service./Run the service.</td></tr><tr><td>5</td><td>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.</td></tr><tr><td></td><td>The exit can pass any valid return and reason code in the first two parameters of the service's parameter list. "Parameters" on page 109 describes the service parameter list.</td></tr><tr><td>6</td><td>CSFSKRC bypass input label parsing./CSFSKRC parse the input label.</td></tr><tr><td>7–31</td><td>Reserved.</td></tr></table>	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 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 service's return and reason codes. See "Return Codes" on page 109 for more information about using exit return codes.	3	Do not call the postprocessing invocation of the service exit./Call the postprocessing invocation of the service exit.	4	Bypass the 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 service's parameter list. "Parameters" on page 109 describes the service parameter list.	6	CSFSKRC bypass input label parsing./CSFSKRC parse the input label.	7–31	Reserved.
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 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 service's return and reason codes. See "Return Codes" on page 109 for more information about using exit return codes.																							
3	Do not call the postprocessing invocation of the service exit./Call the postprocessing invocation of the service exit.																							
4	Bypass the 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 service's parameter list. "Parameters" on page 109 describes the 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 service. For a description of the SPB, see "Secondary parameter block" on page 107.																						
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 200.																						

Table 13. EXPB Control Block Format for Services (continued)

Offset (Dec)	Number of Bytes	Description
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.

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 service. Table 14 describes the contents of SPB.

Table 14. 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 03.
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 200.
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 service.

Table 14. 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: 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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>ICSF was entered via SVC entry from a PCF compatibility macro.</td></tr><tr><td>1</td><td>Original caller in AMODE(31).</td></tr><tr><td>2</td><td>Original caller in AR mode.</td></tr><tr><td>3</td><td>Original caller in SRB mode.</td></tr><tr><td>4</td><td>Original caller in supervisor state or system key.</td></tr><tr><td>5</td><td>Original caller in AMODE(64).</td></tr><tr><td>6–7</td><td>Reserved.</td></tr></table> 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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>The service is using a “storage access” crypto instruction.</td></tr><tr><td>1</td><td>ICSF local lock should be freed when recovery is entered.</td></tr><tr><td>2</td><td>The recovery routine should not retry.</td></tr><tr><td>3</td><td>An I/O subtask has been posted for a dynamic CKDS service call.</td></tr><tr><td>4</td><td>I/O subtask posted for PKDS.</td></tr><tr><td>5</td><td>NQAP in progress.</td></tr><tr><td>6–7</td><td>Reserved.</td></tr></table> Byte 4—Flag byte 2 <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>The service parameter list has a position for a return code.</td></tr><tr><td>1</td><td>The service parameter list has a position for a reason code.</td></tr><tr><td>2</td><td>Internal call to ICSF service.</td></tr><tr><td>3</td><td>SPBLATCH held for CCP array and not for CAMQ.</td></tr><tr><td>4</td><td>Cache operation in process.</td></tr><tr><td>5–7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	ICSF was entered via 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	Original caller in AMODE(64).	6–7	Reserved.	Bit	Meaning When Set On	0	The service is using a “storage access” crypto instruction.	1	ICSF local lock should be freed when recovery is entered.	2	The recovery routine should not retry.	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.	Bit	Meaning When Set On	0	The service parameter list has a position for a return code.	1	The 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	Cache operation in process.	5–7	Reserved.
Bit	Meaning When Set On																																															
0	ICSF was entered via 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	Original caller in AMODE(64).																																															
6–7	Reserved.																																															
Bit	Meaning When Set On																																															
0	The service is using a “storage access” crypto instruction.																																															
1	ICSF local lock should be freed when recovery is entered.																																															
2	The recovery routine should not retry.																																															
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.																																															
Bit	Meaning When Set On																																															
0	The service parameter list has a position for a return code.																																															
1	The 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	Cache operation in process.																																															
5–7	Reserved.																																															
20	4	Protected storage pointer.																																														
24	4	Protected storage length.																																														
28	4	EDC buffer pointer.																																														
32	4	EDC buffer length.																																														

Table 14. 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	8	Original caller's register 1.
60	4	Address of CPRB request storage.
64	4	Length of CPRB request storage.
68	4	Address of CPRB reply storage.
72	4	Length of CPRB reply storage.
76	4	CCPS address.
80	8	RTM token.
88	4	Reserved.

Parameters

Each 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 service parameter list in Register 1. For a description of each service's parameter list, refer to *z/OS Cryptographic Services 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 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 service ignores any return or reason code from the exit. The application program receives the codes from the 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 Cryptographic Services 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 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 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 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.

Note: The cryptographic key data set entry retrieval installation exit will not be given control if SYSPLEXCKDS(YES,FAIL(xxx)) is specified in the ICSF installation options data set.

Environment of the exit

The exit receives control with these 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 this 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 32 describes the installation options data set in further detail. The EXIT keyword has this 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 15 describes the parameters for the CKDS entry retrieval exit.

Table 15. The CKDS Entry Retrieval Exit Parameters

Parameter	Description
1	The address of the current CKDS record. See Table 23 on page 164 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 <i>z/OS Cryptographic Services ICSF Application Programmer's Guide</i> . 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 "Secondary parameter block" on page 107 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 7, “Migration from PCF to z/OS ICSF,” on page 143 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 these 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 this 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 32. The EXIT keyword has this 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 these 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 89 gives a complete list of the registers on entry to the conversion program exit.

Table 16 describes the contents of the exit control block.

Table 16. 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 16. 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: <table><tr><th>Return Code</th><th>Description</th></tr><tr><td>0</td><td>Normal.</td></tr><tr><td>4</td><td>Do not process the entry.</td></tr><tr><td>8</td><td>End conversion program.</td></tr></table>	Return Code	Description	0	Normal.	4	Do not process the entry.	8	End conversion program.		
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Conversion preprocessing invocation.</td></tr><tr><td>1</td><td>Conversion postprocessing invocation.</td></tr><tr><td>2</td><td>Record processing invocation.</td></tr><tr><td>3-7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	Conversion preprocessing invocation.	1	Conversion postprocessing invocation.	2	Record processing invocation.	3-7	Reserved.
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 these 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 these 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 these 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 this 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 32. The EXIT keyword has this 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 89 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 17 describes the RWXP control block.

Table 17. 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: <table><tr><th>Return Code</th><th>Description</th></tr><tr><td>0</td><td>Process current CKDS record</td></tr><tr><td>4</td><td>Do not process current CKDS record</td></tr><tr><td>8</td><td>End processing</td></tr></table>	Return Code	Description	0	Process current CKDS record	4	Do not process current CKDS record	8	End processing
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 17. 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 these 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. These 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 these exit points to control access to the keys in the in-storage CKDS and to the 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.

Next 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. The security service exit (CSFESECS) and security key exit (CSFESECK) can be used to control access to resources on ICSF and for logging in SMF the results of any authorization checks that are made. The security initialization exit defined in the options data set is only invoked if CSFESECS, CSFESECK, or both are also defined.

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 service, before calling any other installation exit that is associated with that service. You can use this exit to control access to a service. Refer to Table 11 on page 101 for a list of 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 these 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 these 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 this 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 this 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 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 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 18 describes the parameters the exit receives:

Table 18. 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 19 describes the parameters this exit receives:

Table 19. 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 9 on page 104 and Table 13 on page 104.

Return codes

All the security exits can pass back a return code in register 15. The security initialization exit supports these 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 these 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 these 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 these 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 Cryptographic Services 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 124 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 these 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 this 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 32. The EXIT keyword has this 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:

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 89 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 20 describes the KGXP control block.

Table 20. 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.

Table 20. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description												
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.												
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. <table><tr><td>0</td><td>Normal, continue processing.</td></tr><tr><td>4</td><td>Reject control statement, but do not end KGUP.</td></tr><tr><td>8</td><td>End KGUP immediately.</td></tr></table> KGUP pre- or postprocessing. <table><tr><td>0</td><td>Normal, continue processing.</td></tr><tr><td>> 0</td><td>End KGUP immediately.</td></tr></table>	0	Normal, continue processing.	4	Reject control statement, but do not end KGUP.	8	End KGUP immediately.	0	Normal, continue processing.	> 0	End KGUP immediately.		
0	Normal, continue processing.													
4	Reject control statement, but do not end KGUP.													
8	End KGUP immediately.													
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>KGUP preprocessing invocation.</td></tr><tr><td>1</td><td>KGUP postprocessing invocation.</td></tr><tr><td>2</td><td>Record preprocessing invocation.</td></tr><tr><td>3</td><td>Record postprocessing invocation.</td></tr><tr><td>4-7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	KGUP preprocessing invocation.	1	KGUP postprocessing invocation.	2	Record preprocessing invocation.	3	Record postprocessing invocation.	4-7	Reserved.
Bit	Meaning When Set On													
0	KGUP preprocessing invocation.													
1	KGUP postprocessing invocation.													
2	Record preprocessing invocation.													
3	Record postprocessing invocation.													
4-7	Reserved.													

Table 20. KGXP Control Block Format (continued)

Offset (Dec)	Number of Bytes	Description																		
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>LABEL with multiple values specified.</td></tr><tr><td>1</td><td>RANGE specified.</td></tr><tr><td>2</td><td>KEY specified.</td></tr><tr><td>3</td><td>CLEAR specified.</td></tr><tr><td>4</td><td>SINGLE specified.</td></tr><tr><td>5</td><td>NOCV specified.</td></tr><tr><td>6</td><td>OUTTYPE specified.</td></tr><tr><td>7</td><td>Reserved.</td></tr></table>	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.
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.																			
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>ADD</td></tr><tr><td>1</td><td>UPDATE</td></tr><tr><td>2</td><td>DELETE</td></tr><tr><td>3</td><td>RENAME</td></tr><tr><td>4</td><td>SET</td></tr><tr><td>5</td><td>OPKYLOAD</td></tr><tr><td>6–7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	ADD	1	UPDATE	2	DELETE	3	RENAME	4	SET	5	OPKYLOAD	6–7	Reserved.		
Bit	Meaning When Set On																			
0	ADD																			
1	UPDATE																			
2	DELETE																			
3	RENAME																			
4	SET																			
5	OPKYLOAD																			
6–7	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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Non-odd parity key was imported.</td></tr><tr><td>1–7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	Non-odd parity key was imported.	1–7	Reserved.												
Bit	Meaning When Set On																			
0	Non-odd parity key was imported.																			
1–7	Reserved.																			

Table 20. 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 79 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 this 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 this 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 20. 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 this 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 this 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 20. 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 20. 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 <i>z/OS Cryptographic Services ICSF Administrator's Guide</i>.
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 20. 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 79 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 132 describes how to use the field in greater detail.

Table 20. 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 Cryptographic Services 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 6. Installation-Defined Callable Services

This topic contains Programming Interface information.

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 Cryptographic Services 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 topic describes how to perform these 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 "Secondary parameter block" on page 107 for a description of the SPB.

The service receives control with these characteristics.

- Supervisor state
- Key 0
- APF authorized
- TCB or SRB mode
- Cross memory mode
- AR mode
- AMODE(31) or AMODE(64)

- 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 10 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 CELLPOL 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 10. 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.

Security access control checking

For the ICSF-defined services, ICSF performs security access control checking to determine if the caller is authorized to access the service and the results of the authorization check can be logged in SMF. This checking is not performed by ICSF for installation-defined services or UDXs. Any security access control checking must be performed by the installation-defined service or UDX itself.

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 this 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 “Steps to create the Installation Options Data Set” on page 21 for more information about ICSF installation options.

The SERVICE keyword has this 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 these 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.

This 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:

- 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 CSFAPRPC, the system calls the callable service that corresponds to the service number in register 0. “The Cryptographic Communication Vector Table (CCVT)” on page 200 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 this 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.

Example of a Service Stub

Figure 11 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
*
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

```

Figure 11. Example of a Service Stub (Part 1 of 5)

```

        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
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

```

Figure 11. Example of a Service Stub (Part 2 of 5)

```

ENDCICS    LR    R15,RC
           L     R14,SAVE14(,R13)
           LM    R1,R12,24(R13)
           BR    R14

*
EDNOCICS   DS    0H
           LR    R15,RC
           LA    R7,12
           CR    R15,R7
           BNE   ENDSVC
           LA    R7,16
           CR    R0,R7
           BNE   ENDSVC
           L     R7,RETURN_CODE_PTR(,R1)
           ST    R15,RETURN_CODE(,R7)
           L     R7,REASON_CODE_PTR(,R1)
           ST    R0,REASON_CODE(,R7)
ENDSVC     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
           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

```

Figure 11. Example of a Service Stub (Part 3 of 5)

```

* 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
*
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
*

```

Figure 11. Example of a Service Stub (Part 4 of 5)

```

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'
          IKJTCTB
          TITLE 'DSECT CCVT'
          CSFCCVT
*
          LA R7,12
          CR R15,R7
          BNE ENDGSVC
          LA R7,16
          CR R0,R7
          BNE ENDGSVC
          L R7,RETURN_CODE_PTR(,R1)
          ST R15,RETURN_CODE(,R7)
          L R7,REASON_CODE_PTR(,R1)
          ST R0,REASON_CODE(,R7)
          ENDGSVC DS 0H

          END

```

Figure 11. Example of a Service Stub (Part 5 of 5)

In Figure 11 on page 137, the service stub, CSFGEN, checks that ICSF is active, places the service number 50 into register 0, and calls CSFAPRPC.

The service number 50 (in the case of this example) 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. 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 “Steps to create the Installation Options Data Set” on page 21 for information about creating the installation options data set and “Changing parameters in the installation options data set” on page 32 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 147 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 145.

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 z/OS 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 147 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 Cryptographic Services 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.

These topics 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 Cryptographic Services 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 Cryptographic Services 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 "Steps to create the CKDS" on page 15 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 these 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 5, "Installation Exits," on page 87 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:

- 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 21 on page 150 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 21. 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 these 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 these 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 these 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.
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 these values: • Blank (same as N) • N • Y
38	1	This field must be blank.

Table 21. Format of Records in the Override File (continued)

Column	Length	Description
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.

These 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 this 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.

These 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.

These 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 Cryptographic Services 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 Cryptographic Services 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.

This 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 these 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 Cryptographic Services 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.

This 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 149 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 Cryptographic Services 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.

Example of a Conversion Initial Activity Report

Figure 12 on page 157 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 12. 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 listed under message CSFV0026 in *z/OS Cryptographic Services 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.

Example of a Conversion Update Activity Report

Figure 13 on page 159 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 Source Entries	12	TOTAL Converted Entries	2	TOTAL ICSF Input Entries	28
				+ 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 13. 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 8. 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.

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 topic gives a brief overview of 4753-HSP and ICSF coexistence considerations. See “Migrating from 4753-HSP” on page 55 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 information.)

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.

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 Cryptographic Services 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 these topics show the format of each of these records.

Format of the CKDS Header Record

Table 22 presents the format of the CKDS header record.

Table 22. 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 <i>yyyymmdd</i> .										
80	8	Creation time	The initial time the CKDS was created in the format <i>hhmmssstth</i> .										
88	8	Last update date	The most recent date the CKDS was updated, in the format <i>yyyymmdd</i> .										
96	8	Last update time	The most recent time the CKDS was updated, in the format <i>hhmmssstth</i> .										
104	2	Sequence number	Initially zero in binary. Incremented each time the data set is processed.										
106	2	CKDS header flag bytes	<table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>The master key verification pattern is valid.</td></tr><tr><td>1</td><td>The master key authentication pattern is valid.</td></tr><tr><td>2–7</td><td>Reserved.</td></tr><tr><td colspan="2">Note: After the bits are set on, the given values remain constant in ICSF.</td></tr></table>	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.	
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.										
116	8	Master key authentication pattern	The system master key authentication pattern.										

Table 22. Cryptographic Key Data Set Header Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
124	72	<i>Reserved</i>	The field is set to binary zeros.
196	52	<i>Installation data</i>	Installation data associated with the CKDS record, as supplied by an installation exit.
248	4	<i>Authentication code</i>	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 23 presents the format of each data set record.

Table 23. Cryptographic Key Data Set Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	64	<i>Key label</i>	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	<i>Key type</i>	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	<i>Creation date</i>	The initial date the CKDS record was created in the format <i>yyyymmdd</i> .
80	8	<i>Creation time</i>	The initial time the CKDS record was created in the format <i>hhmmssst</i> .
88	8	<i>Last update date</i>	The most recent date the CKDS record was updated in the format <i>yyyymmdd</i> .
96	8	<i>Last update time</i>	The most recent time the CKDS record was updated in the format <i>hhmmssst</i> .
104	64	<i>Key token</i>	The internal key token. A key token contains the key value. Refer to "Format of the DES Internal Key Token" on page 166 for the format of the internal key token.

Table 23. Cryptographic Key Data Set Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description										
168	2	CKDS flag bytes	Flag bytes. <table><thead><tr><th>Bit</th><th>Meaning When Set On</th></tr></thead><tbody><tr><td>0</td><td>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.</td></tr><tr><td>1</td><td>Cryptographic key token (CKT) delete.</td></tr><tr><td>2</td><td>CKDS label must be unique.</td></tr><tr><td>3–7</td><td>Reserved.</td></tr></tbody></table> Note: When bit 0 is off, the key within the key token field (offset 104) is an entire key.	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.
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.												
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 AES Internal Key Token

Table 24 shows the format for an AES internal key token.

Table 24. 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'04')						
5	Reserved						
6	Flag byte <table> <tr> <th>Bit</th><th>Meaning When Set On</th></tr> <tr> <td>0</td><td>Encrypted key and master key verification pattern (MKVP) are present. (For a clear AES key token this bit will always be 0.)</td></tr> <tr> <td>1</td><td>Control vector (CV) value in this token has been applied to the key. (For a clear AES key token this bit will always be 0.)</td></tr> </table>	Bit	Meaning When Set On	0	Encrypted key and master key verification pattern (MKVP) are present. (For a clear AES key token this bit will always be 0.)	1	Control vector (CV) value in this token has been applied to the key. (For a clear AES key token this bit will always be 0.)
Bit	Meaning When Set On						
0	Encrypted key and master key verification pattern (MKVP) are present. (For a clear AES key token this bit will always be 0.)						
1	Control vector (CV) value in this token has been applied to the key. (For a clear AES key token this bit will always be 0.)						
7	1-byte LRC checksum of clear key value.						
8–15	Master key verification pattern (MKVP) (For a clear AES key token this value will be hex zeros.)						
16–47	128-bit, 192-bit, or 256-bit key value, left-justified and padded on the right with hex zeros.						

Table 24. Internal Key Token Format (continued)

Bytes	Description
48–55	8-byte control vector. (For a clear AES key token this value will be hex zeros.)
56–57	2-byte integer specifying the length in bits of the clear key value.
58–59	2-byte integer specifying the length in bytes of the encrypted key value. (For a clear AES key token this value will be hex zeros.)
60–63	Token validation value (TVV)

Format of the DES Internal Key Token

Table 25 shows the format for a DES internal key token.

Table 25. 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 <table> <tr> <th>Bit</th><th>Meaning When Set On</th></tr> <tr> <td>0</td><td>Encrypted key and master key verification pattern (MKVP) are present.</td></tr> <tr> <td>1</td><td>Control vector (CV) value in this token has been applied to the key.</td></tr> <tr> <td>2</td><td>Key is used for no control vector (NOCV) processing. Valid for transport keys only.</td></tr> <tr> <td>3</td><td>Key is an ANSI key-encrypting key (AKEK).</td></tr> <tr> <td>4</td><td>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).</td></tr> <tr> <td>5</td><td>AKEK is partially notarized.</td></tr> <tr> <td>6</td><td>Key is an ANSI partial key.</td></tr> <tr> <td>7</td><td>Export prohibited.</td></tr> </table>	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.
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')																		
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')																		

Table 25. Internal Key Token Format (continued)

Bytes	Description
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.

Format of the Clear Key Token: Table 26 shows the format for a clear internal key token.

Table 26. Internal Clear 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 <table> <tr> <th>Bit</th><th>Meaning When Set On</th></tr> <tr> <td>0</td><td>Encrypted key and master key verification pattern (MKVP) are present. This will be off for clear keys.</td></tr> <tr> <td>1</td><td>Control vector (CV) value in this token has been applied to the key. This will be off for clear keys.</td></tr> <tr> <td>2–7</td><td>reserved</td></tr> </table>	Bit	Meaning When Set On	0	Encrypted key and master key verification pattern (MKVP) are present. This will be off for clear keys.	1	Control vector (CV) value in this token has been applied to the key. This will be off for clear keys.	2–7	reserved
Bit	Meaning When Set On								
0	Encrypted key and master key verification pattern (MKVP) are present. This will be off for clear keys.								
1	Control vector (CV) value in this token has been applied to the key. This will be off for clear keys.								
2–7	reserved								
7-15	Reserved (X'00')								
16–23	A single-length key, the left half of a double-length key, or Part A of a triple-length 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.								
32–47	Reserved for clear key tokens (X'00's')								

Table 26. Internal Clear Key Token Format (continued)

Bytes	Description
48–55	X'0000000000000000' if a single-length key or double-length key, or Part C of a triple-length operational key.
56–58	Reserved (X'000000')
59 bits 0 and 1	B'00' reserved
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).

DES External Key Token

Table 27 shows the format for a DES external key token.

Table 27. Format of External Key Tokens

Bytes	Description						
0	X'02' (flag indicating an external key token)						
1	Reserved (X'00')						
2–3	Implementation-dependent bytes (X'0000' for ICSF)						
4	Key token version number (X'00' or X'01')						
5	Reserved (X'00')						
6	Flag byte <table> <tr> <th>Bit</th><th>Meaning When Set On</th></tr> <tr> <td>0</td><td>Encrypted key is present.</td></tr> <tr> <td>1</td><td>Control vector (CV) value has been applied to the key.</td></tr> </table> Other bits are reserved and are binary zeros.	Bit	Meaning When Set On	0	Encrypted key is present.	1	Control vector (CV) value has been applied to the key.
Bit	Meaning When Set On						
0	Encrypted key is present.						
1	Control vector (CV) value has been applied to the key.						
7	Reserved (X'00')						
8–15	Reserved (X'0000000000000000')						
16–23	Single-length key or left half of a double-length key, or Part A of a triple-length key. The value is encrypted under a transport key.						
24–31	X'0000000000000000' if a single-length key or right half of a double-length key, or Part B of a triple-length key. The right half of a double-length key or Part B of a triple-length key is encrypted under a transport (key-encrypting key) for export or import.						
32–39	Control vector (CV) for single-length key or left half of CV for double-length key						
40–47	X'0000000000000000' if single-length key or right half of CV for double-length key						
48–55	X'0000000000000000' if a single-length key, double-length key, or Part C of a triple-length key.						
56–58	Reserved (X'000000')						
59 bits 0 and 1	B'00'						
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 (see “Token Validation Value” on page 167 for a description).						

DES Null Key Token

Table 28 shows the format for a DES null key token.

Table 28. Format of Null Key Tokens

Bytes	Description
0	X'00' (flag indicating this is a null key token).
1–15	Reserved (set to binary zeros).
16–23	Single-length encrypted key, or left half of double-length encrypted key, or Part A of triple-length encrypted key.
24–31	X'0000000000000000' if a single-length encrypted key, the right half of double-length encrypted key, or Part B of triple-length encrypted key.
32–39	X'0000000000000000' if a single-length encrypted key or double-length encrypted key.
40–47	Reserved (set to binary zeros).
48–55	Part C of a triple-length encrypted key.
56–63	Reserved (set to binary zeros).

End of Programming Interface information

Public Key Data Set (PKDS) Format

The PKDS record includes the PKDS header and the PKA key token. These tables show the format of each of these records.

Format of the PKDS Header Record

Table 29. Public Key Data Set Header Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	64	<i>PKHVKEY</i>	VSAM key of the PKDS header.
64	8		Reserved.
72	8	<i>PKHCRDTE</i>	The date the PKDS was created in the format <i>yyyymmdd</i> .
80	8	<i>PKHCRTIM</i>	The initial time the PKDS was created in the format <i>hhmmssstth</i> .
88	8	<i>PKHUPDTE</i>	The most recent date the PKDS header was updated, in the format <i>yyyymmdd</i> .
96	8	<i>PKHUPTIM</i>	The most recent time the PKDS header was updated, in the format <i>hhmmssstth</i> .
104	4	<i>PKHRLN</i>	Length of the PKDS header entry.
108	16	<i>PKHKMKHP</i>	The hash pattern of the KMMK.
124	16	<i>PKHSMKHP</i>	The hash pattern of the SMK.
140	20		Reserved.
160	20	<i>PKHAUTH</i>	PKDS header authentication code.

Format of the PKDS Record

Table 30. Public Key Data Set Record Format

Offset (Dec)	Number of Bytes	Field Name	Description
0	64	<i>PKDLABEL</i>	Label or name of this PKDS entry.
64	8		Reserved.
72	8	<i>PKDCRDTE</i>	The date this PKDS record was created in the format <i>yyyymmdd</i> .
80	8	<i>PKDCRTIM</i>	The initial time this PKDS record was created in the format <i>hhmmssst</i> .
88	8	<i>PKDUPDTE</i>	The most recent date this PKDS record was updated, in the format <i>yyyymmdd</i> .
96	8	<i>PKDUPTIM</i>	The most recent time this PKDS record was updated, in the format <i>hhmmssst</i> .
104	4	<i>PKDRLEN</i>	Length of the entire PKDS record entry.
108	52	<i>PKDUDATA</i>	User data.
160	20	<i>PKDAUTH</i>	The entry authentication code.
180	1868	<i>PKDTOKEN</i>	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 31 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 31. 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.
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 512-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 32 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 32. 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.

Table 32. DSS Public Key Token (continued)

Offset (Dec)	Number of Bytes	Description
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. $y = g^x \text{ mod}(p)$; $1 < y < p$.

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 4096 bits for use with the PCICC, PCIXCC or CEX2C.
- 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 33 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 33. 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.

Table 33. RSA Private External Key Token Basic Record Format (continued)

Offset (Dec)	Number of Bytes	Description
RSA Private Key Section (required)		
- For 1024-bit Modulus-Exponent form refer to “RSA Private Key Token, 1024-bit Modulus-Exponent External Form” - For 4096-bit Chinese Remainder Theorem form refer to “RSA Private Key Token, 4096-bit Chinese Remainder Theorem External Form” on page 174		
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 512-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 34. 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'.

Table 34. RSA Private Key Token, 1024-bit Modulus-Exponent External Format (continued)

Offset (Dec)	Number of Bytes	Description						
050	004	Key use flag bits. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Key management usage permitted.</td></tr><tr><td>1</td><td>Signature usage not permitted.</td></tr></table> All other bits reserved, set to binary zero.	Bit	Meaning When Set On	0	Key management usage permitted.	1	Signature usage not permitted.
Bit	Meaning When Set On							
0	Key management usage permitted.							
1	Signature usage not permitted.							
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, 4096-bit Chinese Remainder Theorem External

Form: This RSA private key token (up to 2048-bit modulus) is supported on the PCICC, PCIXCC or CEX2C. The 4096-bit modulus private key token is supported on the z9 EC, z9 BC and System z10 Enterprise Class with the Nov 2007 or later version of the licensed internal code installed on the CEX2C.

Table 35. RSA Private Key Token, 4096-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 + qqg + rrr + sss + uuu + 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'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'.

I Table 35. RSA Private Key Token, 4096-bit Chinese Remainder Theorem External Format (continued)

Offset (Dec)	Number of Bytes	Description
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: qq.
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: uu.
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	qq	Prime number, q
132 + ppp + qq	rrr	$d_p = d \bmod(p - 1)$
132 + ppp + qq + rrr	sss	$d_q = d \bmod(q - 1)$
132 + ppp + qq + rrr + sss	uu	$U = q^{-1} \bmod(p)$.
132 + ppp + qq + rrr + sss + uu	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.	
132 + ppp + qq + rrr + sss + uu + xxx	nnn	Modulus, n. $n = pq$ where p and q are prime and $1 < n < 2^{4096}$.

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 36 on page 176 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 36. 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.
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.

Table 36. DSS Private External Key Token (continued)

Offset (Dec)	Number of Bytes	Description
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. $y = g^x \text{ mod}(p)$ 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 37. 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.

Table 37. RSA Private Internal Key Token Basic Record Format (continued)

Offset (Dec)	Number of Bytes	Description																		
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 179 - For 1024-bit X'06' Modulus-Exponent form refer to “RSA Private Key Token, 1024-bit Modulus-Exponent Internal Form for PCICC, PCIXCC or CEX2C” on page 179 - For 4096-bit X'08' Chinese Remainder Theorem form refer to “RSA Private Key Token, 4096-bit Chinese Remainder Theorem Internal Form” on page 181																				
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 512 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'.																		
004	004	PKA token type. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>RSA key.</td></tr><tr><td>1</td><td>DSS key.</td></tr><tr><td>2</td><td>Private key.</td></tr><tr><td>3</td><td>Public key.</td></tr><tr><td>4</td><td>Private key name section exists.</td></tr><tr><td>5</td><td>Private key unenciphered.</td></tr><tr><td>6</td><td>Blinding information present.</td></tr><tr><td>7</td><td>Retained private key.</td></tr></table>	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.
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.																		

Table 37. RSA Private Internal Key Token Basic Record Format (continued)

Offset (Dec)	Number of Bytes	Description
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 38. 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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Key management usage permitted.</td></tr><tr><td>1</td><td>Signature usage not permitted.</td></tr></table> All other bits reserved, set to binary zero.	Bit	Meaning When Set On	0	Key management usage permitted.	1	Signature usage not permitted.
Bit	Meaning When Set On							
0	Key management usage permitted.							
1	Signature usage not permitted.							
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 PCICC, PCIXCC or CEX2C:

Table 39. RSA Private Internal Key Token, 1024-bit ME Form for PCICC, PCIXCC or CEX2C

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.

Table 39. RSA Private Internal Key Token, 1024-bit ME Form for PCICC, PCIXCC or CEX2C (continued)

Offset (Dec)	Number of Bytes	Description						
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Key management usage permitted.</td></tr><tr><td>1</td><td>Signature usage not permitted.</td></tr></table> All other bits reserved, set to binary zeros.	Bit	Meaning When Set On	0	Key management usage permitted.	1	Signature usage not permitted.
Bit	Meaning When Set On							
0	Key management usage permitted.							
1	Signature usage not permitted.							
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. $d=e^{-1}mod((p-1)(q-1))$, 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.						
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, 4096-bit Chinese Remainder Theorem Internal Form:

This RSA private key token (up to 2048-bit modulus) is supported on the PCICC, PCIXCC or CEX2C. The 4096-bit modulus private key token is supported on the z9 EC, z9 BC and System z10 Enterprise Class with the Nov 2007 or later version of the licensed internal code installed on the CEX2C.

Table 40. RSA Private Internal Key Token, 4096-bit Chinese Remainder Theorem Internal 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 + qqg + rrr + sss + uuq + +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: <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Key management usage permitted.</td></tr><tr><td>1</td><td>Signature usage not permitted.</td></tr></table> All other bits reserved, set to binary zero.	Bit	Meaning When Set On	0	Key management usage permitted.	1	Signature usage not permitted.
Bit	Meaning When Set On							
0	Key management usage permitted.							
1	Signature usage not permitted.							
054	002	Length of prime number, p, in bytes: ppp.						
056	002	Length of prime number, q, in bytes: qqg.						
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: uuq.						
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.						

Table 40. RSA Private Internal Key Token, 4096-bit Chinese Remainder Theorem Internal Format (continued)

Offset (Dec)	Number of Bytes	Description
132 + ppp	qqq	Prime number, q
132 + ppp + qqg	rrr	$d_p = d \bmod(p - 1)$
132 + ppp + qqg + rrr	sss	$d_q = d \bmod(q - 1)$
132 + ppp + qqg + rrr + sss	uuu	$U = q^{-1} \bmod(p)$.
132 + ppp + qqg + rrr + sss + uuu	ttt	Random number r (used in blinding).
132 + ppp + qqg + rrr + sss + uuu + ttt	iii	Random number r^{-1} (used in blinding).
132 + ppp + qqg + 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 + qqg + rrr + sss + uuu + ttt + iii + xxx	nnn	Modulus, n. $n = pq$ where p and q are prime and $1 < n < 2^{4096}$.

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 41 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 41. 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'.

Table 41. DSS Private Internal Key Token (continued)

Offset (Dec)	Number of Bytes	Description
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.
012	002	Size of the y field in bytes, "yyy".
014	yyy	Public key, y . $y = g^x \text{ mod}(p)$; 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)		

Table 41. DSS Private Internal Key Token (continued)

Offset (Dec)	Number of Bytes	Description												
000	004	Eye catcher 'PKTN'.												
004	004	PKA token type. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>RSA key.</td></tr><tr><td>1</td><td>DSS key.</td></tr><tr><td>2</td><td>Private key.</td></tr><tr><td>3</td><td>Public key.</td></tr><tr><td>4</td><td>Private key name section exists.</td></tr></table>	Bit	Meaning When Set On	0	RSA key.	1	DSS key.	2	Private key.	3	Public key.	4	Private key name section exists.
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.												

Format of the PKA Trusted Block Key Token

The trusted block key token is an extension of the PKA key token, using new section identifiers. The trusted block key token contains the following optional section identifiers:

- X'11'
- X'12'
- X'13'
- X'15'

Section identifier X'14' is required.

The sections may be in any order. TLV Subsections may be in any order within their section.

Table 42 presents the format of a trusted block 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 42. Trusted Block Token

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier (a flag that indicates token type): • 0X'1E' = External token. MAC key in Trusted Block Information section is protected by a variant of the IMP-PKA transport KEK. • 0X'1F' = Internal token. MAC key in Trusted Block Information section is protected by the PKA master key.
1	1	Header structure version number = 0X'00'.
2	2	Length of the trusted block token.
4	4	Reserved, binary zeros.
Trusted Public Key Section (optional)		

Table 42. Trusted Block Token (continued)

Offset (Dec)	Number of Bytes	Description
0	1	0X'11', section identifier, Trusted RSA Public Key.
1	1	Structure version number 0X'00'.
2	2	Length of this section, in bytes = 16+x+y.
4	2	Reserved, binary zeros.
6	2	RSA Public Key Exponent field length in bytes, "x.
8	2	RSA Public Key Modulus length in bits.
10	2	RSA Public Key Modulus field length in bytes, "y.
12	x	Public Key Exponent, e.
12+x	y	Public Key Modulus, n.
12+x+y	4	Flags: 0X'00000000' - public key can be used in digital signature operations only 0X'C0000000' - public key can be used in key management operations only 0X'80000000' - public key can be used in both digital signature and key management operations Notes: - All other bits are reserved and must be zero - For DSV, flags must be set to either 0X'00000000' or 0X'80000000' - For RKX asymmetric envrypted key output operations, flags must be set to either 0X'C0000000' or 0X'80000000'
Trusted Block Information Section (required)		
0	1	0X'14', section identifier, Trusted Block Information.
1	1	Structure version number 0X'00'.
2	2	Length of this section, in bytes = 10 + x.
4	2	Reserved, binary zeros.
6	4	Flags: 0X'00000000' Trusted Block is in the Inactive state 0X'00000001' Trusted Block is in the Activated state
10	x	TLV objects for optional fields. the TLV object for the Protection Information is required.
Trusted Block Label Section (optional)		
0	1	0X'13', Section identifier, Trusted Block Label
1	1	Structure version number 0X'00'
2	2	Length of this section, 68
4	64	Label. Host software can use this label to perform access control checking, for example with RACF.
Application Defined Data Section (optional)		
0	1	0X'15', Section identifier, Application Defined Data
1	1	Structure version number 0X'00'
2	2	Length of this section = x+6 (max value is 406)
4	2	Length Application Defined Data (between 0 and 400 inclusive)

Table 42. Trusted Block Token (continued)

Offset (Dec)	Number of Bytes	Description
6	x	Application defined data may be used to hold a public key certificate for the trusted public key. The data is variable length but must not cause the entire trusted block token to exceed its maximum allowed size of 2500 bytes.
Rule Section (optional)		
0	1	0X'12', Section identifier, Rules
1	1	Structure version number 0X'00'
2	2	Length of this section = 20+y
4	8	Rule ID, ASCII value identifying the rule. The rule ID for this rule must be different from the ID used in any other rule section in this trusted block. A rule ID used in any other rule section in this trusted block. A rule ID must be an 8-byte ASCII character string with no null terminator. The acceptable character set is: A-Z, a-z, 0-9, - (hyphen), and _ (underscore). All other characters are reserved for future use. If the Rule ID is less than 8 characters long, it must be left justified and padded on the right with space characters (i.e. 0X'20').
12	4	Flags (All other bits are reserved and must be zero): • bit0 (LSB) Generate / Export, If '0', generate new key. If '1', export existing key.
16	1	Generate Key Length - length in bytes of key to be generated if rule is for key gen (flags @ offset 12, bit0='0').
17	1	Key Check Algorithm Identifier (All other values are reserved and must not be used). • 0X'00' - Do not compute a key check value. Set the key check output length to zero. • 0X'01' - Encrypt an 8-byte block of binary zeros with the key. If the key is single-length, the DEA key encodes the 8 bytes. If the key is double-length, the key DEA triple-encodes the 8-byte value as follows: the left half key encodes the zero-bit value, this result is DEA decoded by the right key half, and that result is DEA encoded by the left key half. If the key is single-length, return the leftmost 3 or 4 bytes of the result of the encryption according to the value of the key_check_length input parameter as follows: – If the key is single-length and if the key_check_length is set to 3, 3 bytes will be returned. – If the key is single-length and if the key_check_length is set to a value between 4 and 8 bytes inclusive, 4 bytes will be returned. – If the key is longer than single-length, return the entire 8 byte encryption result. – Under no circumstances will there be a 1, 2, 5, 6, or 7 byte encryption result returned. • 0X'02' - Compute the MDC2 hash of the key and return the 16 byte value.

Table 42. Trusted Block Token (continued)

Offset (Dec)	Number of Bytes	Description
18	1	Symmetric Encrypted Output Key Format Flag - Indicates output symmetric key formats allowed by this rule • 0X'00' - Output as an RKX token encrypted under a variant of the MAC key. • 0X'01' - Output as a CCA token encrypted under a transport key. The input parameter <code>transport_key_identifier</code> may be an RKX token, CCA token, or label of a CCA token. A transport key variant (in the Transport Key Variant TLV) and/or a CV (in the Common Export Key Parameters) may be applied to the transport key. If both are applied, the variant is applied first, then the CV. Note: All other values are reserved.
19	1	Asymmetric Encrypted Output Key Format - Indicates output symmetric key formats allowed by this rule • 0X'00' - No asymmetric key should be output. • 0X'01' - Output in PKCS1.2 format. • 0X'02' - Output in RSAOAEP format. This format can make use of Extra Data provided as an input parameter to RKX. Note: All other values are reserved.
20	y	TLV Objects for optional fields. This is a series of zero or more objects in TLV format.

Trusted Block Information Subsections:

Table 43. Protection Information TLV Object

Offset (Dec)	Number of Bytes	Description
Protection Information TLV Object (required)		
0	2	Tag, 0X'0001'
2	2	Length of this TLV = 62
Values for Protection Information TLV Object		
0	1	Structure version number 0X'00'
1	1	Reserved, binary zeros
2	32	Encrypted MAC Key
34	8	MAC. ISO 16609 TDES CBC MAC value.
42	16	MKVP for the PKA master key, computed using MDC4. This is set to binary zeros if the trusted block is external.

Table 44. Activation and Expiration Data TLV Object

Offset (Dec)	Number of Bytes	Description
Activation and Expiration Data TLV Object (optional)		
0	2	Tag, 0X'0002'
2	2	Length of this TLV = 16
Values for Activation and Expiration Data TLV Object		
0	1	Structure version number 0X'00'
1	1	Reserved, binary zeros

Table 44. Activation and Expiration Data TLV Object (continued)

Offset (Dec)	Number of Bytes	Description
2	2	Flags: 0X'0000' Activation and Expiration dates should not be checked by the card (dates in this TLV are ignored) 0X'0001' Activation and Expiration dates are checked by the card, comparing them to the card's internal RTC
4	4	Activation date for trusted block. Layout: Big-endian year (2 bytes), month (1 byte,) day of month (1 byte)
8	4	Expiration date for trusted block. Layout: Big-endian year (2 bytes), month (1 byte,) day of month (1 byte)

Rule Subsections:

Table 45. Transport Key Variant TLV Object

Offset (Dec)	Number of Bytes	Description
Transport Key Variant TLV Object (optional)		
0	2	Tag, 0X'0001'
2	2	Length of this TLV = 8+n
Values for Transport Key Variant TLV Object		
0	1	Structure version number 0X'00'
1	2	Reserved, binary zeros
3	1	Length of Variant field starting at offset 4, in bytes = n.
4	n	Variant to be XORed into the cleartext transport key. The length must be greater than or equal to the length of the transport key that is supplied to RKX when this rule is used. If the variant is longer than the transport key, it is truncated on the right to the length of the key before it is used. This variant is used if the Length of Variant at offset 3 is non-zero AND if the Symmetric Encrypted Output Key Format Flag (offset 018) = 0X'01' (CCA Token Format). Note: Transport keys are not used for output RKX tokens.

Transport Key Rule Reference TLV Object should be present if an application programmer intends to use an RKX token as a transport key, otherwise the RKX transport key will not be accepted for use.

Table 46. Transport Key Rule Reference TLV Object

Offset (Dec)	Number of Bytes	Description
Transport Key Rule Reference TLV Object (optional)		
0	2	Tag, 0X'0002'
2	2	Length of this TLV = 14
Values for Transport Key Rule Reference TLV Object		
0	1	Structure version number 0X'00'
1	1	Reserved, binary zeros
2	8	Rule ID for the rule that must have been used to create the transport key.

Common Export Key Parameters TLV Object must be present if exporting an existing key in either RKX token or CCA token format.

Table 47. Common Export Key Parameters TLV Object

Offset (Dec)	Number of Bytes	Description
Common Export Key Parameters TLV Object (required for all key export)		
0	2	Tag, 0X'0003'
2	2	Length of this TLV = 12+x+y
Values for Common Export Key Parameters TLV Object		
0	1	Structure version number 0X'00'
1	2	Reserved, binary zeros
3	1	Flags (reserved and must be set to binary zeros)
4	1	Export Key Minimum Length, in bytes
5	1	Export Key Maximum Length, in bytes (this value is used as the "y" in the length column in this TLV Object and in the CCA Token Export Key Parameters TLV Object below).
6	1	Output Key Variant Length, in bytes "x". Note: The Output Key Variant is not used if this field is set to binary zero.
7	x	Output Key Variant, which will be XORed with the cleartext value of the key being exported. This must be at least as long as the longest key that will be exported using the rule. If the variant is longer than the key, the leftmost bytes of the variant are used, up to the key length (e.g. If the variant is 24 bytes and the key is 16 bytes, the first 16 bytes of the variant are XORed with the key and the remaining 8 bytes of the variant are not used).
7+x	1	Length of CV field starting at offset 8+x, in bytes "y". Note: The CV is not used if this field is set to binary zero.

Table 47. Common Export Key Parameters TLV Object (continued)

Offset (Dec)	Number of Bytes	Description
8+x	y	CV to be XORed with cleartext transport key - A zero length at offset 7+x will have the following effect: If the source key is a CCA token, its CV will be set when encrypting the source key under the transport key. This CV will be preserved in the symmetric encrypted output if the output is to be in the form of a CCA token. - If the CV length at offset 7+x is non-zero, then the CV length must be greater than or equal to the length of the key contained in the source_key_identifier. Currently, the CV length can only be 8 or 16 bytes. - If the CV length at offset 7+x is 16, and the source key is single length. - If the CV length at offset 7+x is 16, and this CV is equal to 16 bytes of: - all the CV bit definitions are ignored, - the symmetric encrypted output key (if CCA token format) will contain a Flags value (offset 6) set to indicate "CV present, and the CV (at offset 32-47) in the output key will contain 16 binary zeroes, - If the key contained in the source_key_identifier input parameter is 8 bytes in length, the key will not be replicated to 16 bytes. - If the access control point 0X'00DB' is active ("Replicate Key Allowed) and if the source_key_identifier input parameter contains an 8-byte key in either CCA or RKX key token format; and if this CV is 16 bytes in length (both halves non-zero), then the key contained in the source_key_identifier input parameter is replicated (equal halves) if the key-form bits "fff in this CV do not specify 000 (single-length key) AND the left most "f bit (bit 40) is not set to '1. - If the Symmetric Encrypted Output Key Format Flag = 0X'00' (RKX Token Format), then this CV is ignored. Note: Transport keys are not used for output symmetric RKX tokens. - A transport key with equal left and right halves may not be used to encrypt a source key.

Export Key Rule Reference TLV Object must be present if using an RKX token as an export key, otherwise the RKX export key will not be accepted for use.

Table 48. Export Key Rule Reference TLV Object

Offset (Dec)	Number of Bytes	Description
Export Key Rule Reference TLV Object (required for key export of RKX tokens)		
0	2	Tag, 0X'0004'
2	2	Length of this TLV = 14
Values for Export Key Rule Reference TLV Object		
0	1	Structure version number 0X'00'
1	1	Reserved, binary zeros
2	8	Rule ID for the rule that must have been used to create the export key.

CCA Token Export Key Parameters TLV Object must be present if using a CCA token as an export key, otherwise the CCA export key will not be accepted for use.

Table 49. CCA Token Export Key Parameters TLV Object

Offset (Dec)	Number of Bytes	Description
CCA Token Export Key Parameters TLV Object (required for key export of CCA tokens)		
0	2	Tag, 0X'0005'
2	2	Length of this TLV = 10+2y+z
Values for CCA Token Export Key Parameters TLV Object		
0	2	Structure version number 0X'00'
1	2	Reserved, binary zeros
3	1	Flags (all bits are reserved and must be set to binary zeros)
4	1	Export Key CV Limit Mask Length, in bytes. If CV limits to be used (Flags bit 0='1'), then CV Limit Mask length "y must equal the Export Key Maximum Length in Common Export Key Parameters TLV Tag 0X'0003'. If CV limits are not to be used, then the CV Limit Mask length "y must = 0. Examples: - An Export Key Minimum Length of 24 bytes and a CV Limit Mask Length of 16 is not permitted. - If CV limits are not to be used, then either the CV Limit Mask length "y must = 0, or this TLV Object must be absent from the trusted block. - The CV Limit Mask Length must be equal in length to the actual length of the source_key input parameter to the RKX service.
5	y	Export Key CV Limit Mask (this field is nonexistent if y = 0).
5+y	y	Export Key CV Limit Template (length is equal to that of the CV limit mask) - This field is nonexistent if y=0 - The CV limit mask and CV limit template have the same length, which is the length defined in the field Export Key Maximum Length. If the key to be exported is less than this length, the CV limit mask length will be longer than the actual key, which results in a rejection of the request with an error. - Two fields work together to restrict the acceptable control vectors for CCA keys to be exported. - One field contains a mask (CV Limit Mask) to indicate which CV bits should be checked. The second field contains the required values for those CV bits that are to be checked (CV Limit Template). The CV for the key to be exported is first ANDed with the CV Limit Mask, and then the result is compared with the CV Limit Template. If the two do not match, the request is rejected with an error. - Using the mask and template to permit multiple key types cannot consistently be achieved with one Rule section. For example, setting the CV mask and template bit10 to '1 will permit PIN keys and CVAR keys; however, finding a mask and template to permit KEKs and PIN keys is not possible. It is therefore recommended that one rule section (and thus one Export Key CCA Token Parms TLV Object) be used to check the limits for one particular key type, e.g. DATA keys, KEKs, or PIN keys. A trusted block with multiple rule sections (and thus multiple Export Key CCA Token Parms TLV Objects) can be used to permit multiple key types to be exported with RKX.

Table 49. CCA Token Export Key Parameters TLV Object (continued)

Offset (Dec)	Number of Bytes	Description
5+2y	1	Source Key Label Template Length, in bytes ("z = 0 or 64, no other value allowed") - If this length = 0, and a source key label is present, the label value will not be checked. - If this length = 64, and a source key label is not present, an error will be returned by the RKX service. - A single asterisk (wildcard) in the Source Key Label Template field will not be interpreted to accept the absence of a source key label.
6+2y	z	Source Key Label Template (this field is nonexistent if z = 0). Label templates are left-justified within this field. Examples of valid values include: *, A*, *A, *AB*. The Label template must conform to the following rules: - First character cannot be between 0X'00' - 0X'1F', nor can it be 0X'FF'. - First character cannot be numeric, 0X'30' - 0X'39'. - A label is terminated by a space on the right. The remainder of the 64-byte field is padded with space characters (0X'20'). - The only special characters permitted are 0X'23' (#), 0X'24' (\$), 0X'40' (@), and wildcard 0X'2A'. - The wildcard 0X'2A' (*) is permitted as the first character, the last character, or only character. - Once a space is encountered, every character to the right up to the 64B length must be spaces. - Only alphanumeric characters are allowed a-z, A-Z, 0-9, the 3 special characters, the wildcard (*), and the space (0X'20').

End of Programming Interface information

Token data set (TKDS) format

A z/OS PKCS #11 token represents a virtual cryptographic device, and can contain multiple objects. The token data set (TKDS) contains definitions of z/OS PKCS #11 tokens and token objects.

The token data set includes a header record and records for each of the individual z/OS PKCS #11 tokens and token objects. Each object associated with a particular z/OS PKCS #11 token has the token's name in its handle. The records are variable length records, and contain a length field specifying the total length of the record.

Format of the header record of the token data set

There is one header record for the token data set.

Table 50. Format of the header record of the token data set

Offset (decimal)	Length of field (bytes)	Description
0	72	VSAM key of the TKDS header Bytes 0-39: Binary zeros Bytes 40-43: EBCDIC "THDR" Bytes 44-71: Binary zeros
72	8	Reserved for IBM's use

Table 50. Format of the header record of the token data set (continued)

Offset (decimal)	Length of field (bytes)	Description
80	8	The date that the TKDS was created, in the format <i>yyyymmdd</i>
88	8	The time that the TKDS was created, in the format <i>hhmmssst</i>
96	8	The most recent date that the TKDS header was updated, in the format <i>yyyymmdd</i>
104	8	The most recent time that the TKDS header was updated, in the format <i>hhmmssst</i>
112	4	Length of the TKDS header record
116	40	Reserved for IBM's use

Format of the token and object records

Each z/OS PKCS #11 token record and token object record begins with the same 188 bytes of data. The remainder of the record is specific to the token or object.

Common section of the token and object records

Every record in the token data set, with the exception of the header record, begins with these 188 bytes of data.

Table 51. Format of the common section of the token and object records

Offset (decimal)	Length of field (bytes)	Description
0	72	Handle of token or object Bytes 0-31: Token name Bytes 32-39: Sequence number Byte 40: Character "T" for token object Bytes 41-43: Blank characters Bytes 44-71: Binary zeros
72	8	Reserved for IBM's use
80	8	The date that this record was created, in the format <i>yyyymmdd</i>
88	8	The time that this record was created, in the format <i>hhmmssst</i>
96	8	The most recent date that this record was updated, in the format <i>yyyymmdd</i>
104	8	The most recent time that this record was updated, in the format <i>hhmmssst</i>
112	4	Length of the entire TKDS record entry
116	20	Reserved for IBM's use
136	52	User data
188	variable	The TKDS token or object (see mappings)

Format of the token-specific section of the token record

Each z/OS PKCS #11 token record begins with the 188 bytes. The remainder of the record contains the contents of the token. The mapping of the record shows the data beginning at offset 0, which is its offset into the token-specific portion of the

record; however, that portion of the record is at an offset of 188 into the entire record.

Table 52. Format of the unique section of the token record

Offset (decimal) 188 +	Length of field (bytes)	Description
0	4	Eye catcher for token: "TOKN"
4	2	Version number of structure: EBCDIC '00'
6	2	Length of structure in bytes
8	4	Reserved for IBM's use. Must be zeros.
12	8	Last assigned sequence number
20	32	Manufacturer identification
52	16	Model
68	16	Serial number
84	8	Date of the most recent update to this token, expressed as Coordinated Universal Time (UTC) in the format <i>yyyymmdd</i> . This includes any update to token information or to a token object.
92	8	Time of the most recent update to this token, expressed as Coordinated Universal Time (UTC) in the format <i>hhmmssst</i> . This includes any update to token information or to a token object.
100	44	Reserved for IBM's use
144		End of token

Format of the object-specific sections of the token object records

There are five classes of objects that can be associated with a z/OS PKCS #11 token:

- Certificate
- Public key
- Private key
- Secret key
- Data

The token object record for each begins with the common section described "Common section of the token and object records" on page 193, followed by a section specific to the class of object. Each of the object-specific sections begins with a 12-byte header record, followed by a variable-length section. Each 12-byte header contains a 4-byte flag field that has the same mapping for all classes of objects.

Table 53. Format of the token object flags. This 4-byte flag field occurs in the object header section of each token object record.

Offset (decimal)	Field name	Description
Flag byte 1		
Bit 0	OBJ_IS_TOKOBJ	When on, the object is a token object. When off, the object is a session object.

Table 53. Format of the token object flags (continued). This 4-byte flag field occurs in the object header section of each token object record.

Offset (decimal)	Field name	Description
Bit 1	OBJ_IS_PRVOBJ	When on, the object is a private object. When off, the object is a public object.
Bit 2	OBJ_IS_MODOBJ	When on, the object is modifiable.
Bit 3	KEY_DERIVE	When on, the key supports key derivation.
Bit 4	KEY_LOCAL	When on, the key was generated locally.
Bit 5	KEY_ENCRYPT	When on, the key supports encryption.
Bit 6	KEY_DECRYPT	When on, the key supports decryption.
Bit 7	KEY_VERIFYA	When on, the key supports verification where the signature is an appendix to the data.
Flag byte 2		
Bit 0	KEY_VERIFYR	When on, the key supports verification where the data is recovered from the signature
Bit 1	KEY_SIGA	When on, the key supports signatures where the signature is an appendix to the data.
Bit 2	KEY_SIGR	When on, the key supports signatures where the data is recovered from the signature.
Bit 3	KEY_WRAP	When on, the key supports wrapping.
Bit 4	KEY_UNWRAP	When on, the key supports unwrapping.
Bit 5	KEY_EXTRACT	When on, the key is extractable.
Bit 6	KEY_IS_SENSITIVE	When on, the key is sensitive.
Bit 7	KEY_IS_ALWAYS_SENSITIVE	When on, the SENSITIVE attribute (KEY_IS_SENSITIVE) is always true.
Flag byte 3		
Bit 0	KEY_NEVER_EXTRACT	When on, the EXTRACTABLE attribute (KEY_EXTRACT) is never true. When off, the EXTRACTABLE attribute (KEY_EXTRACT) can be true.
Bit 1	OBJ_IS_TRUSTED	When on, the certificate can be trusted for the application for which it was created.
Bit 2	CERT_IS_DEFAULT	When on, this is the default certificate.
Bits 3-7		Reserved for IBM's use
Flag byte 4		
Bits 0-7		Reserved for IBM's use

Table 54. Format of the token certificate object

Offset (decimal) 188 +	Length of field (bytes)	Description
Object header		
0	4	Eye catcher for certificate object: "CERT"
4	2	Version: EBCDIC '00'
6	2	Length of the object (in bytes)
8	4	Flags (see Table 53 on page 194)
Object type-specific section		
12	4	TYPE attribute: X'00000000': CKC_X_509
16	4	Certificate category 0 Undefined 1 Token user 2 Certificate authority 3 Other entity
20	8	Reserved for IBM's use
28	32	Reserved for IBM's use
60	2	Length of SUBJECT attribute in bytes (<i>aa</i>)
62	2	Length of ID attribute in bytes (<i>bb</i>)
64	2	Length of ISSUER attribute in bytes (<i>cc</i>)
66	2	Length of SERIAL_NUMBER attribute in bytes (<i>dd</i>)
68	2	Length of VALUE attribute in bytes (<i>ee</i>)
70	2	Length of LABEL attribute in bytes (<i>ff</i>)
72	2	Length of APPLICATION attribute in bytes (<i>gg</i>)
74	22	Reserved for IBM's use
96	4	Offset of SUBJECT attribute in bytes
100	4	Offset of ID attribute in bytes
104	4	Offset of ISSUER attribute in bytes
108	4	Offset of SERIAL_NUMBER attribute in bytes
112	4	Offset of VALUE attribute in bytes
116	4	Offset of LABEL attribute in bytes
120	4	Offset of APPLICATION attribute in bytes
124	44	Reserved for IBM's use
168	<i>aa + bb + cc + dd + ee + ff + gg</i>	Certificate attributes (variable length)
<i>168 + aa + bb + cc + dd + ee + ff + gg</i>		End of certificate object

Table 55. Format of the token public key object

Offset (decimal) 188 +	Length of field (bytes)	Description
Object header		
0	4	Eye catcher for public key object: "PUBK"

Table 55. Format of the token public key object (continued)

Offset (decimal) 188 +	Length of field (bytes)	Description
4	2	Version: EBCDIC '00'
6	2	Length of the object (in bytes)
8	4	Flags (see Table 53 on page 194)
Object type-specific section		
12	4	TYPE attribute: CKK_RSA
16	8	Start date for the key, in the format <i>yyyymmdd</i>
24	8	End date for the key, in the format <i>yyyymmdd</i>
32	4	Key generate mechanism: CK_UNAVAILABLE_INFORMATION
36	36	Reserved
72	4	Length in bits of modulus <i>n</i>
76	256	Modulus <i>n</i>
332	256	Reserved
588	256	Public exponent <i>e</i>
844	256	Reserved
1100	2	Length of SUBJECT attribute in bytes (<i>aa</i>)
1102	2	Length of ID attribute in bytes (<i>bb</i>)
1104	2	Length of LABEL attribute in bytes (<i>cc</i>)
1106	2	Length of APPLICATION attribute in bytes (<i>dd</i>)
1108	20	Reserved
1128	4	Offset of SUBJECT attribute in bytes
1132	4	Offset of ID attribute in bytes
1136	4	Offset of LABEL attribute in bytes
1140	4	Offset of APPLICATION attribute in bytes
1144	40	Reserved
1184	<i>aa+bb+cc+dd</i>	Public key attributes (variable length)
1184+ <i>aa+bb+cc+dd</i>		End of public key object

Table 56. Format of the token private key object

Offset (decimal) 188 +	Length of field (bytes)	Description
Object header		
0	4	Eye catcher for private key object: "PRIV"
4	2	Version: EBCDIC '00'
6	2	Length of object (in bytes)
8	4	Flags (see Table 53 on page 194)
Object type-specific section		
12	4	Type attribute: CKK_RSA
16	8	Start date for the key (in the format <i>yyyymmdd</i>)

Table 56. Format of the token private key object (continued)

Offset (decimal) 188 +	Length of field (bytes)	Description
24	8	End date for the key (in the format <i>yyyymmdd</i>)
32	4	Key generate mechanism: CK_UNAVAILABLE_INFORMATION
36	36	Reserved
72	4	Length in bits of modulus <i>n</i>
76	256	Modulus: modulus <i>n</i>
332	256	Reserved
588	256	Public exponent <i>e</i>
844	256	Reserved
1100	32	Reserved
1132	256	Private exponent <i>d</i>
1388	256	Reserved
1644	136	Prime <i>p</i>
1780	128	Reserved
1908	128	Prime <i>q</i>
2036	128	Reserved
2172	136	Private exponent <i>d</i> modulo <i>p</i> -1
2300	128	Reserved
2428	128	Private exponent <i>d</i> modulo <i>q</i> -1
2556	128	Reserved
2684	136	CRT coefficient <i>q</i> -1 mod <i>p</i>
2820	128	Reserved
2948	2	Length of SUBJECT attribute in bytes (<i>xx</i>)
2950	2	Length of ID attribute in bytes (<i>yy</i>)
2952	2	Length of LABEL attribute in bytes (<i>zz</i>)
2954	2	Length of APPLICATION attribute in bytes (<i>ww</i>)
2956	20	Reserved
2976	4	Offset of SUBJECT attribute in bytes
2980	4	Offset of ID attribute in bytes
2984	4	Offset of LABEL attribute in bytes
2988	4	Offset of APPLICATION attribute in bytes
2992	40	Reserved
3032	<i>xx+yy+zz+ww</i>	Private key attributes (variable length)
3032+ <i>xx+yy+zz+ww</i>		End of private key object

Table 57. Format of the token secret key object

Offset (decimal) 188 +	Length of field (bytes)	Description
Object header		
0	4	Eye catcher for secret key object: "SECK"

Table 57. Format of the token secret key object (continued)

Offset (decimal) 188 +	Length of field (bytes)	Description
4	2	Version: EBCDIC '00'
6	2	Length of the object in bytes
8	4	Flags (see Table 53 on page 194)
Object type-specific section		
12	4	Type of key X'00000013' CKK_DES X'00000014' CKK_DES2 X'00000015' CKK_DES3 X'0000001F' CKK_AES
16	8	Start date for the key (in the format <i>yyyymmdd</i>)
24	8	End date for the key (in the format <i>yyyymmdd</i>)
32	4	Key generate mechanism CK_UNAVAILABLE_INFORMATION
36	2	Length of the key in bytes
38	32	Reserved
70	64	VALUE: value of the key
134	544	Reserved
678	2	Length of LABEL attribute in bytes (<i>xx</i>)
680	2	Length of APPLICATION attribute in bytes (<i>yy</i>)
682	2	Length of the ID attribute in bytes (<i>zz</i>)
684	20	Reserved
704	4	Offset of LABEL attribute in bytes
708	4	Offset of APPLICATION attribute in bytes
712	4	Offset of the ID attribute in bytes
716	40	Reserved
756	<i>xx+yy+zz</i>	Secret key attributes (variable length)
756+ <i>xx+yy+zz</i>		End of secret key object

Table 58. Format of the token data object

Offset (decimal) 188 +	Length of field (bytes)	Description
Object header		
0	4	Eye catcher for data object: "DATA"
4	2	Version: EBCDIC '00'
6	2	Length of object, in bytes
8	4	Flags (see Table 53 on page 194)
Object type-specific section		
12	4	Reserved for IBM's use
16	28	Reserved for IBM's use
44	2	Length of VALUE attribute in bytes (<i>aa</i>)
46	2	Length of OBJECT_ID attribute in bytes (<i>bb</i>)

Table 58. Format of the token data object (continued)

Offset (decimal) 188 +	Length of field (bytes)	Description
48	2	Length of LABEL attribute in bytes (<i>cc</i>)
50	2	Length of APPLICATION attribute in bytes (<i>dd</i>)
52	2	Length of ID attribute in bytes (<i>ee</i>)
54	22	Reserved for IBM's use
76	4	Offset of VALUE attribute in bytes
80	4	Offset of OBJECT_ID attribute in bytes
84	4	Offset of LABEL attribute in bytes
88	4	Offset of APPLICATION attribute in bytes
92	4	Offset of ID attribute in bytes
96	44	Reserved for IBM's use
140	$aa + bb + cc + dd + ee$	Data attributes (variable length)
$140 + aa + bb + cc + dd + ee$		End of data object

Data Areas

These topics 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 under the line.

Programming Interface information

CCVT

ONLY these fields are part of the programming interface:

- CCVTDACC
- CCVTCCVE
- CCVTHFLG
- CCVTPRPC
- CCVTINST
- CCVTINS2
- CCVTLNTH
- CCVTFMID
- CCVT_USERPARM

End of Programming Interface information

Table 59 on page 201 describes the contents of the Cryptographic Communication Vector Table.

Table 59. 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	CCVTVR	Version. The version of the CCVT. This field must contain the character string 03																		
6	2	CCVTLEN	Length. The length of the CCVT. The value of this field is 296 in decimal.																		
8	1	CCVTAUX	Auxilliary flags. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>ICSF is terminating.</td></tr></table>	Bit	Meaning When Set On	0	ICSF is terminating.														
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		Reserved																		
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.																		
36	1	CCVTSFG1	Status byte. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>ICSF services are active.</td></tr><tr><td>1</td><td>At least one Integrated Cryptographic Feature has a valid master key.</td></tr><tr><td>2</td><td>ICSF initialization complete.</td></tr><tr><td>3</td><td>ICSF is active and PCF is not active.</td></tr><tr><td>4</td><td>Compatibility is permitted. COMPAT(YES) or COMPAT(COEXIST) is specified.</td></tr><tr><td>5</td><td>At least one Integrated Cryptographic Feature is valid.</td></tr><tr><td>6</td><td>SEC 250 or greater.</td></tr><tr><td>7</td><td>S/390 Enterprise Servers and S/390 Multiprise Cryptographic Coprocessor Feature is in use.</td></tr></table>	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 greater.	7	S/390 Enterprise Servers and S/390 Multiprise Cryptographic Coprocessor Feature is in use.
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 greater.																				
7	S/390 Enterprise Servers and S/390 Multiprise Cryptographic Coprocessor Feature is in use.																				

Table 59. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description																		
37	1	CCVTFLAG	Flag byte. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>VTAM puts terminal buffer greater than 16MB line.</td></tr><tr><td>1</td><td>Hardware environment tested.</td></tr><tr><td>2</td><td>GTMAC opcode in hardware.</td></tr><tr><td>3</td><td>PCI Cryptographic Coprocessor hardware instructions available.</td></tr><tr><td>4</td><td>At least one PCI Cryptographic Coprocessor is active.</td></tr><tr><td>5</td><td>Additional hardware environment tested.</td></tr><tr><td>6</td><td>At least one PCI Cryptographic Coprocessor is online.</td></tr><tr><td>7</td><td>At least one PCI Cryptographic Coprocessor is present.</td></tr></table>	Bit	Meaning When Set On	0	VTAM puts terminal buffer greater than 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.
Bit	Meaning When Set On																				
0	VTAM puts terminal buffer greater than 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.																				
38	1	CCVTOFLG	Operational flag byte. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Configuration is under PR/SM.</td></tr><tr><td>1</td><td>Close the CKDS</td></tr><tr><td>2</td><td>Key record create, key record delete, and key record write disallowed.</td></tr><tr><td>3</td><td>I/O subtask is available.</td></tr><tr><td>4</td><td>CCVT_DEF_ALG bit. If on, CDMF is the system default algorithm; if off, DES is the default.</td></tr><tr><td>5</td><td>CCVT_CDMF_ENA bit. If on, hardware is capable of performing CDMF.</td></tr><tr><td>6</td><td>PKA master keys are valid.</td></tr><tr><td>7</td><td>Use ICSF reason codes.</td></tr></table>	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.
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.																		

Table 59. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description																
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Crypto assist instructions available.</td></tr><tr><td>1</td><td>Additional secure Crypto device available.</td></tr><tr><td>2</td><td>Support for 64-bit callers.</td></tr><tr><td>3</td><td>ICSF Cross-System Services environment is active for CKDS</td></tr><tr><td>4</td><td>ICSF Cross-System Services environment is active for TKDS</td></tr><tr><td>5</td><td>RSA 4096-bit function enabled and the RNGL service is available</td></tr><tr><td>6-7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	Crypto assist instructions available.	1	Additional secure Crypto device available.	2	Support for 64-bit callers.	3	ICSF Cross-System Services environment is active for CKDS	4	ICSF Cross-System Services environment is active for TKDS	5	RSA 4096-bit function enabled and the RNGL service is available	6-7	Reserved.
Bit	Meaning When Set On																		
0	Crypto assist instructions available.																		
1	Additional secure Crypto device available.																		
2	Support for 64-bit callers.																		
3	ICSF Cross-System Services environment is active for CKDS																		
4	ICSF Cross-System Services environment is active for TKDS																		
5	RSA 4096-bit function enabled and the RNGL service is available																		
6-7	Reserved.																		
81	1	CCVTSFLG	Flag bytes. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>ICSF during initialization.</td></tr><tr><td>1-7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	ICSF during initialization.	1-7	Reserved.										
Bit	Meaning When Set On																		
0	ICSF during initialization.																		
1-7	Reserved.																		
82	2		Reserved.																
84	4	CCVTENF	ECB for ENF listen.																
88	4	CCVTTCB	ICSF maintask TCB address.																
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.																

Table 59. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description																		
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>KMMK is valid for CP0.</td></tr><tr><td>1</td><td>SMK is valid for CP0.</td></tr><tr><td>2</td><td>KMMK has been reset for CP0.</td></tr><tr><td>3</td><td>SMK has been reset for CP0.</td></tr><tr><td>4</td><td>KMMK is valid for CP1.</td></tr><tr><td>5</td><td>SMK is valid for CP1.</td></tr><tr><td>6</td><td>KMMK has been reset for CP1.</td></tr><tr><td>7</td><td>SMK has been reset for CP1.</td></tr></table>	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.
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	<table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0 and 1</td><td>SMK status for KSU0.</td></tr><tr><td>2 and 3</td><td>KMMK status for KSU0.</td></tr><tr><td>4 and 5</td><td>SMK status for KSU1.</td></tr><tr><td>6 and 7</td><td>KMMK status for KSU1.</td></tr></table>	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.								
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.																		
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. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>PKDS available.</td></tr><tr><td>1</td><td>Signal PKDS I/O to close PKDS.</td></tr><tr><td>2</td><td>At least one PCICA is active.</td></tr><tr><td>3-7</td><td>Reserved.</td></tr></table>	Bit	Meaning When Set On	0	PKDS available.	1	Signal PKDS I/O to close PKDS.	2	At least one PCICA is active.	3-7	Reserved.								
Bit	Meaning When Set On																				
0	PKDS available.																				
1	Signal PKDS I/O to close PKDS.																				
2	At least one PCICA is active.																				
3-7	Reserved.																				
197	1	CCVTCICS	CICS processing flags. <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>CSFAPRPD installed.</td></tr><tr><td>1</td><td>CSFACKWL installed.</td></tr></table>	Bit	Meaning When Set On	0	CSFAPRPD installed.	1	CSFACKWL installed.												
Bit	Meaning When Set On																				
0	CSFAPRPD installed.																				
1	CSFACKWL installed.																				

Table 59. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
198	1	CCVTYAFF	Bit Meaning When Set On 0 ZKA compliance environment.
199	1	CSFTTKDF	TKDS processing flags Bit Meaning When Set On 0 TKDS available 1 Signal TKDS I/O to close TKDS 2-7 Reserved
200	4	CCVTPRPD	Address of CSFAPRPD.
204	4	CCVTCKWL	Address of CSFACKWL.
208	4	CCVTPC4	PC4 (CSFMCAMP) number.
212	4	CCVTPERR	Error routine for POST macro.
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 ENF2 listen exit.
232	4	CCVTENA3	Address of ENF3 listen exit.
236	4	CCVTPC5	PC5 (CSFMCCPP entry).
240	4	CCVTPC6	PC6 (CSFMWCFS entry).
244	16	CCVT_KXMD	Hardware feature status. Bit Meaning When Set On 0 Reserved. 1 SHA-1 enabled. 2 SHA-256 enabled. 3 SHA-512 enabled. 4-7 Reserved. Bytes 2–16 are 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 59. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description	
280	16	CCVT_KMC_EXPORT	Hardware feature status.	
			Bit	Meaning When Set On
			0	Reserved.
			1	KMC DES enabled.
			2	Reserved.
			3	KMC TDES enabled.
			4-17	Reserved.
			18	KMC AES 128 key enabled.
			19	KMC AES 192 key enabled.
			20	KMC AES 256 key enabled.
			21-23 Reserved. Change bytes 4-16 are reserved.	
296	4	CCVTPC7	PC7 (CSFMGARM entry)	
300	4	CCVTPC8	PC8 (CSFMGTRM entry)	
304	8	CCVTGART	Token of CSFMGARC resource manager	
312	8	CCVTGTRT	Token of CSFMGTRC resource manager	
320	4	CCVTGARC	Address of CSFMGARC resource manager	
324	4	CCVTGTRC	Address of CSFMGTRC resource manager	
328	4	CCVT_IDENTITY	Identifier	
332	6	CCVT_GARCDATE	Compile date of CSFMGARC (EBCDIC)	
338	6	CCVT_GTRCDATE	Compile date of CSFMGTRC (EBCDIC)	
344	4	CCVTEPRP	Address of CSFEPRPC	
348	4	CCVTPKB6	Address of CSFSPKB6	
352	4	CCVTVRET	Address of CSFVRET	
356	4	CCVTWRET	Address of CSFWRET	
360	4	CCVTENA4	Address of ENF4 listen exit	
364	4	CCVTENF4	ENF Token for sysplex member leave/join event	
368	4	CCVTSPXE	ECB to post for sysplex member leave/join event	
372	4	CCVT_CCSTCB	Address of TCB of of Cross-System Services task	
376	4	CCVTCSS	Address of Cross-Services Block	
380	4	CCVTTIOE	ECB for TKDS I/O subtask	
384	4	CCVTTIOC	ECB for TKDS I/O work complete	
388	4	CCVTTIOA	Address of ASCB task posting the TKDS I/O subtask	
392	4	CCVTIDNR TKDS	Subtask called ID for TKDS	
396	4	*	Reserved	
400	4	CCVTTPLEX_ECB	ECB to post for TKDS sysplex member leave/join event	
404	4	CCVTPPLEX_ECB	ECB to post for PKDS sysplex member leave/join event	
408	4	CCVT_CSST_TCB	Address of Cross-System Services TCB for TKDS	

Table 59. Cryptographic Communication Vector Table (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
412	4	CCVTCSSST	Address of Cross-System Services block for TKDS sharing
416	4	CCVT_CSSP_TCB	Address of Cross-System Services TCB for PKDS
420	4	CCVTCSSP	Address of Cross-System Services block for PKDS sharing
424	4	CCVTTDSA	Address of beginning of TKDS data space
428	4	CCVTTDST	ALET of TKDS data space
432	8	CCVTTDSK	STOKEN of TKDS data space
440	4	CCVTDSR	ECB to post for data space management requests
444	4	CCVTDSC	ECB to post for data space management work complete
448	4	CCVTDSAS	ASCB address for non-CSF address
452	28	*	Reserved
480		*	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 these fields are part of the programming interface:

- CCVEINPP
- CCVEINPL
- CCVESECC

End of Programming Interface information

Table 60 describes the contents of the Cryptographic Communication Vector Table Extension.

Table 60. 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 60. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description																																																		
16	4	CCVESTAT	Status word First status byte – CCVESTA1 <table><tr><th>Bit</th><th>Meaning When Set On</th></tr><tr><td>0</td><td>Special secure mode allowed.</td></tr><tr><td>1</td><td>Special secure mode enabled.</td></tr><tr><td>2</td><td>ICSF is in test mode.</td></tr><tr><td>3</td><td>Authentication required for key retrieval.</td></tr><tr><td>4</td><td>The hardware has gone from active to inactive.</td></tr><tr><td>5</td><td>First start of ICSF during this IPL.</td></tr><tr><td>6</td><td>Security Server (RACF) checking required for authorized callers.</td></tr><tr><td>7</td><td>PCF coexistence.</td></tr></table> Second status byte – CCVESTA2 <table><tr><td>0</td><td>Dynamic CKDS updates disallowed.</td></tr><tr><td>1</td><td>Refresh needed.</td></tr><tr><td>2</td><td>Dynamic CKDS creates disallowed.</td></tr><tr><td>3</td><td>Linear CKDS 80% full.</td></tr><tr><td>4</td><td>80% message already sent.</td></tr><tr><td>5</td><td>CDMF used (rather than DES). This indicates setting of COMPENC keyword.</td></tr><tr><td>6</td><td>PKA callable services disallowed.</td></tr><tr><td>7</td><td>Authenticate the CKT when bit is one</td></tr></table> Third status byte – CCVESTA3 <table><tr><td>0</td><td>PKDS read not permitted.</td></tr><tr><td>1</td><td>PKDS write, create, and delete not permitted.</td></tr><tr><td>2</td><td>SYSPLEXCKDS(YES) was specified in Install Options Data Set.</td></tr><tr><td>3</td><td>SYSPLEXCKDS(YES,FAIL(YES)) was specified in Install Options Data Set.</td></tr><tr><td>4</td><td>SYSPLEXTKDS(YES) was specified in Install Options Data Set.</td></tr><tr><td>5</td><td>SYSPLEXTKDS(YES,FAIL(YES)) was specified in Install Options Data Set.</td></tr><tr><td>6</td><td>TKDS refresh requested.</td></tr><tr><td>7</td><td>TKDS empty at initialization.</td></tr></table> Remaining 1 byte is reserved.	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.	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	Authenticate the CKT when bit is one	0	PKDS read not permitted.	1	PKDS write, create, and delete not permitted.	2	SYSPLEXCKDS(YES) was specified in Install Options Data Set.	3	SYSPLEXCKDS(YES,FAIL(YES)) was specified in Install Options Data Set.	4	SYSPLEXTKDS(YES) was specified in Install Options Data Set.	5	SYSPLEXTKDS(YES,FAIL(YES)) was specified in Install Options Data Set.	6	TKDS refresh requested.	7	TKDS empty at initialization.
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.																																																				
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	Authenticate the CKT when bit is one																																																				
0	PKDS read not permitted.																																																				
1	PKDS write, create, and delete not permitted.																																																				
2	SYSPLEXCKDS(YES) was specified in Install Options Data Set.																																																				
3	SYSPLEXCKDS(YES,FAIL(YES)) was specified in Install Options Data Set.																																																				
4	SYSPLEXTKDS(YES) was specified in Install Options Data Set.																																																				
5	SYSPLEXTKDS(YES,FAIL(YES)) was specified in Install Options Data Set.																																																				
6	TKDS refresh requested.																																																				
7	TKDS empty at initialization.																																																				

Table 60. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
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 <i>create trace entry</i> routine.
36	4	CCVETRCB	Pointer to the current trace buffer. Bit 0 Meaning When Set On 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 212 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.

Table 60. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
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.
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 <i>Issue ENF SIGNAL</i> .
216	4	CVESMCA	Address of SMCA
220	4	CCVE_SUBPOOL	Subpool for storage
224	4	*	Reserved
228	4	CCVEMKVB	Pointer to the current Master Key Verification Pattern (MKVP) block. See “Master Key Verification Pattern Block (MKVB)” on page 211 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	*	Reserved
344	4	CCVEPMKV	Address of CSFMPMKV.
348	16	*	Reserved
364	4	CCVEWKAR	Work area for services.
368	4	*	Reserved.
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.

Table 60. Cryptographic Communication Vector Table Extension (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
432	4	CCVEIOPB_PKDS	Address of PKDS I/O st data.
436	4	CCVEIOST_TKDS	Address of TKDS I/O subtask TCB
440	4	CCVEIOPB_TKDS	Address of TKDS I/O subtask data
444	44	CCVETKDS	Data set name of TKDS
488	4	CCVEPKDL	Pointer to PKDS interface
492	44	CCVEPKDS	Data set name of the PKDS
536	4	CCVECCPD	Address of CAJ data
540	4	CCVECCPV	Address of private CAJ data
544	4	CCVEGSVT	Address of generic service vector table
548	4	CCVEGSFL	GSVT flags
552	54	CCVEWLDS	Data set name of Wait List data set
606	2	*	Padding
608	4	CCVEMUST	Address of UDX service table
612	4	CCVEQSCC	Queue State Change Count
616	4	CCVEPKCH	Size of PKDS cache in records
620	4	CCVEPCXY	Address of CSFMPCXY
624	4	CCVEPCXH	Address of CSFMPCXH
628	4	CCVEPCXC	Address of CSFMPCXC
632	4	CCVEDSCB	Address of control block for data space management
636	4	CCVETDSL	Pointer to TKDS interface
640	4	CCVETDLL	Pointer to TKDL interface
644	24	*	Reserved
672			Ensure CCVE ends on a doubleword boundary

Master Key Verification Pattern Block (MKVB)

Table 61 describes the contents of the MKVB.

Table 61. 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 62 describes the format of the generic service table, a control block that is used to control the call of installation-defined services.

Table 62. Generic Service Table Block Format

Offset (Dec)	Number of Bytes	Description
0	4	EBCDIC 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.
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 63 describes the contents of the performance measurements for RMF. The count fields are double-word length.

Table 63. 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.

Table 63. RMF Measurements Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
20	8	DACC_ENT_ID	Identifier of count array - character 'ENCSDDES'. 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 ENCSDDES service calls.
36	8	DACC_ENT_BYT_CNT	Count of ENCSDDES bytes processed.
44	8	DACC_ENT_INT_CNT	Count of ENCSDDES 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.
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.

Table 63. RMF Measurements Record Format (continued)

Offset (Dec)	Number of Bytes	Field Name	Description
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.
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.
308	8	DACC_ENT_ID	Identifier of count array - character 'OWH256'. The One Way Hash service will collect data as follows: For SHA-224 and SHA-256, the number of service calls, number of bytes of data hashed, and the number of instructions will be collected.
316	8	DACC_ENT_SVC_CNT	Count of OWH service calls for SHA-224 and SHA-256.
324	8	DACC_ENT_BYT_CNT	Count of OWH bytes processed for SHA-224 and SHA-256.
332	8	DACC_ENT_INT_CNT	Count of OWH instructions for SHA-224 and SHA-256.
340	8	DACC_ENT_ID	Identifier of count array - character 'OWH512'. The One Way Hash service will collect data as follows: For SHA-384 and SHA-512, the number of service calls, number of bytes of data hashed, and the number of instructions will be collected.
348	8	DACC_ENT_SVC_CNT	Count of OWH instructions for SHA-384 and SHA-512.
356	8	DACC_ENT_BYT_CNT	Count of OWH instructions for SHA-384 and SHA-512.
364	8	DACC_ENT_INT_CNT	Count of OWH instructions for SHA-384 and SHA-512.

Appendix B. 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. You must have CICS 4.1 or higher.

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.

Installing the CICS-ICSF Attachment Facility

Before you can use the CICS-ICSF Attachment Facility, the ICSF system programmer, or the CICS administrator needs to install it. This involves:

- 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.

Steps for installing the CICS-ICSF attachment facility

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. This 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 SCSFMODE(CSFATREN)
        ENTRY DFHEAI
NAME CSFATREN(R)
        INCLUDE SDFHLOAD(DFHEAI)
        REPLACE CSFDHEAI(DFHEAI),CSF0EAI
        INCLUDE SCSFMODE(CSFATREU)
        ENTRY DFHEAI
NAME CSFATREU(R)
/*

```

2. Include the ICSF load module data set in the CICS startup job control language as shown in this example.

```

//DFHRPL DD DISP=SHR,DSN=xxxxx.SDFHLOAD
//      DD DISP=SHR,DSN=yyy.SCSFMODE (The ICSF load library)
//      DD ...
...
//SYSIN DD DISP=SHR,DSN=xxxxx.SYSIN(DFH$SIPx)
...

```

In the previous 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.

This 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 CSFATREU)
*
* ----- Delimiter between Stages 2 and 3 -----
*
DFHPLT TYPE=ENTRY,PROGRAM=DFHDELIM
*
* ----- Third stage of initialization -----
* (none)
*
DFHPLT TYPE=FINAL
END
/*

```

The previous 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 these controls:

```

INCLUDE OBJLIB(DFHPLTyy)
NAME DFHPLTyy(R)

```

5. The CICS administrator should customize the system CSD to include:

- CSFATREU
- CSFATREN
- A PLT to indicate that initialization is to call CSFATREN to enable the ICSF TRUE, CSFATREU

This 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 previously.

```
//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(DFHPLTy) 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 this statement:

```
ENABLE PROGRAM('CSFATRUE') TALENGTH(250) LINKEDITMODE START
```

You can also do this in a single step with this statement:

```
CECI ENABLE PROGRAM('CSFATRUE') TALENGTH(250) LINKEDITMODE START
```

6. If you have any existing CICS applications which invoke any of the ICSF services in the Wait List, then these applications must be re-linked with the current ICSF stubs.

Implementing the CICS wait list

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 which can complete asynchronously (for example, those services which perform I/O to a cryptographic key data set and those services which are routed to a PCICC or PCIXCC). If the 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 (for CCF systems with PCICCs) and CSFWTL01 (for systems with PCIXCCs) 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 these conditions.

- Non-CICS customers will not specify a WAITLIST keyword.
- CICS customers who want to use the default CICS Wait List shipped with HCR7720 will not specify a WAITLIST keyword. If you have any existing CICS applications which invoke any of the ICSF services in the Wait List, then these applications must be re-linked with the current ICSF stubs.

- 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 (for CCF systems with PCICCs) or member CSFWTL01 (for systems with PCIXCCs/CEX2Cs) of SYS1.SAMPLIB. The WAITLIST keyword in the Installation Options Data Set should be set to point to this data set. If you have any existing CICS applications which invoke any of the ICSF services in the Wait List, then these applications must be re-linked with the current ICSF stubs.

If you already have the CICS-ICSF Attachment facility installed, there are a number of callable services which may potentially be routed to the PCICC or PCIXCC/CEX2C for processing. If you have a modified CICS Wait List, you should ensure that the wait list data set includes all such services, and any CICS applications which invoke any of these services are re-linked with the current ICSF stubs. As a model, 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 (for CCF systems with PCICCs) or in member CSFWTL01 (for systems with PCIXCCs/CEX2Cs) 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.

For CCF Systems:

- 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 Cryptographic Services 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.

For CCF systems with a PCICC or z990/z890 systems with a PCIXCC/CEX2C:

- 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 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 32 for additional information.

Appendix C. Helpful Hints for ICSF First Time Startup

The purpose of this topic 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, “z890, z990, z9 EC, z9 BC and System z10 Enterprise Class without a PCIXCC or CEX2CC,” on page 233 if you’re running in this environment.

Checklist for First-Time Startup of ICSF

This is a checklist for the first-time startup of ICSF.

Step 1. Hardware Setup - CCF Systems

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	<i>Support Element Operations Guide</i>
Completed	

Step 1. Hardware Setup - PCIXCC/CEX2C Systems

Process	LIC installed for CP Assist for Cryptographic Functions Note: If using TKE V4.0 or higher and your z890/z990 system is running with May 2004 or higher version LIC or your z9 EC or z9 BC system is running with MCL 029 Stream J12220 or higher version of LIC, you must Permit each PCIXCC/CEX2C for TKE Commands.
Responsible	CE or Client Operator Representative
Where	Support Element
Verify	Via CPC details • CP Assist for Cryptographic Functions is 'Installed' • CP Assist for Cryptographic Functions DES/TDES enablement (feature 3863) is 'Installed' Via PCI Cryptographic Configuration Task • Status for each PCICA, PCIXCC/CEX2C , or CEX2A is 'Configured' Note: If using TKE V4.0 or higher, the status for each PCIXCC/CEX2C is 'Permitted' when May 2004 or higher

LIC is installed on a z890/z990 or MCL 029 Stream
J12220 or higher is installed on a z9 EC or z9 BC.

References *Support Element Operations Guide*

Completed

Step 2. LPAR Activation Profiles - CCF Systems

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 and PCICA's that CAN be online • PCI Cryptographics Coprocessor Online List includes all PCICC's and PCICAs 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	<i>Support Element Operations Guide</i> <i>z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide, SA23-2211 (LPAR Considerations)</i> <i>zSeries PR/SM Planning Guide</i>

Completed

Step 2. LPAR Activation Profiles - PCIXCC/CEX2C Systems

Process	PCI Crypto Page Setup
Responsible	CE or Client Operator Representative
Where	Support Element
Verify	Control Domain Index

Usage Domain Index

PCI Cryptographic Candidate List includes all PCIXCCs/CEX2Cs, PCICAs, and CEX2As that CAN be online

PCI Cryptographic Online List includes all PCIXCCs/CEX2Cs, PCICAs, and CEX2As that WILL be online when activation is complete (Selections in the Online List MUST be selected in the Candidate List)

References *Support Element Operations Guide*
z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide, SA23-2211 (LPAR Considerations)
zSeries PR/SM Planning Guide

Completed

Note: If TKE is to be used, then ALL PCIXCCs/CEX2Cs that you want TKE to be able to control MUST be defined in the Online and Candidate Lists. Also, the Usage Domain for the TKE Host LPAR **MUST** be unique. While the same domain may be used by other LPARs as long as these LPARs do not share any of the same PCIXCC/CEX2C cards, the TKE Host domain must have access to all the PCIXCC/CEX2C cards so that prohibits any other LPAR from using the same domain.

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 LNKLIST concatenation • Update PROGxx to APF authorize CSF.SCSFMOD0 • Update IKJTSOxx for ICSF by adding CSFDAUTH and CSFDPKDS to the AUTHPGM and AUTHTSF parameter lists CKDS and PKDS created ICSF Startup Procedure created Installation Options Dataset created • 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) Note: SSM(YES) is not required with a PCIXCC/CEX2C Access provided to the ICSF panels
References	Chapter 2, "Installation, Initialization, and Customization," on page 13
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
	Perform passphrase or smart card setup
	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)
References	CSFTTKE profile is defined in the RACF FACILITY and RACF APPL classes
	<i>z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide, SA23-2211 (See Topics: TKE Workstation Setup and Customization and TKE TCP/IP and Host Considerations)</i>
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 13
Completed	

Step 6. Loading Master Keys and Initializing the CKDS through ICSF Panels

Note: If sharing a CKDS between CCF systems and non-CCF systems, the CKDS must be initialized on a CCF system.

If you are using TKE, proceed to the next step.

Process	Passphrase Initialization to load and SET master keys and initialize CKDS and PKDS
	• Create NOCV, ANSI, and ESYS keys as applicable for your installation
Note: These system keys are not valid on a PCIXCC/CEX2C system.	

- 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 (CCF, PCICC and PCIXCC/CEX2C) 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 - CCF systems only
- Initialize the PKDS
- 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 (CCF Systems):

```
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)
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY SERVICES ARE NOW AVAILABLE
```

In System Log (PCIXCC/CEX2C Systems):

```
CSFM416I BOTH MASTER KEYS CORRECT ON PCI X CRYPTOGRAPHIC
COPROCESSOR Xnn, SERIAL NUMBER nnnnnnnnn.
CSFM431I BOTH MASTER KEYS CORRECT ON CRYPTO EXPRESS2
COPROCESSOR Epp, SERIAL NUMBER nnnnnnnnn.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY SERVICES ARE NOW AVAILABLE
CSFM440I PCI X CRYPTOGRAPHIC COPROCESSOR Xpp,
SERIAL NUMBER nnnnnnnnn, ACTIVE.
CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp,
SERIAL NUMBER nnnnnnnnn, ACTIVE.
```

Message CSFM440I will be issued for each active PCIXCC.

Message CSFM441I will be issued for each active CEX2C.

In System Log (CEX2C without CEX2A Systems):

```
S CSF
CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.
CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2 COPROCESSOR Epp,
SERIAL NUMBER nnnnnnnnn.
CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.
CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.
```

```
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn,
ACTIVE.
```

Message CSFM434E will be issued for each CEX2C online.

Message CSFM441I will be issued for each active CEX2C.

In System Log (CEX2C and CEX2A Systems):

```
S CSF
CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.
CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2
COPROCESSOR Epp,SERIAL NUMBER nnnnnnnn.
CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE
CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn,
ACTIVE.
```

Message CSFM434E will be issued for each CEX2C online.

Message CSFM435I will be issued for each active CEX2A.

Message CSFM441I will be issued for each active CEX2C.

In System Log (CPACF only (no CEX2C or CEX2A) System):

```
S CSF
CSFM507I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC COPROCESSORS ONLINE.
CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
```

In System Log (CPACF and CEX2A)

```
S CSF
CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE
CSFM507I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC COPROCESSORS ONLINE.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
```

Message CSFM435I will be issued for each active CEX2A.

References *z/OS Cryptographic Services ICSF Administrator's Guide*,
SA22-7521 (See Topics: Using the Pass Phrase Initialization Utility
and Managing Master Keys on the zSeries Enterprise Server)

Completed

Step 7. Customizing TKE and Loading Master Keys

If you are not using TKE, proceed to the next step.

Process - CCF Systems

TKE Administrator's and Key Officers

- Define Host IDs
- 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.

Process - PCIXCC/CEX2C Systems

TKE Administrator's and Key Officers

- Define Host IDs
- Define Roles
- Define PCIXCC/CEX2C Authorities
- Load New SYM-MK
- Load and SET New ASYM-MK

Note: If you have more than one PCIXCC/CEX2C, repeat the process for each, unless Groups have been defined.

Responsible ICSF Administrator

- Initialize CKDS and SET the DES/SYM-MK New Master Key
- Create NOCV, ANSI, and ESYS keys as applicable for your installation - CCF Systems only
- Load PKA/ASYM-MK Master Keys
- SET ASYM-MK (PCICC/PCIXCC/CEX2C)
- Initialize the PKDS
- Enable PKA Services
- Enable PKDS Read Access
- Enable PKDS Write, Create, and Delete Access

Where TKE Workstation and ICSF Panels

Verify In System Log (CCF Systems):

```
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)
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY SERVICES ARE NOW AVAILABLE
```

In System Log (PCIXCC/CEX2C Systems):

```
CSFM416I BOTH MASTER KEYS CORRECT ON PCI X CRYPTOGRAPHIC
COPROCESSOR Xnn, SERIAL NUMBER nnnnnnnnn.
CSFM431I BOTH MASTER KEYS CORRECT ON CRYPTO EXPRESS2
COPROCESSOR Epp, SERIAL NUMBER nnnnnnnnn.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY SERVICES ARE NOW AVAILABLE
CSFM440I PCI X CRYPTOGRAPHIC COPROCESSOR Xpp,
SERIAL NUMBER nnnnnnnnn, ACTIVE.
CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp,
SERIAL NUMBER nnnnnnnnn, ACTIVE.
```

Message CSFM440I will be issued for each active PCIXCC.

Message CSFM441I will be issued for each active CEX2C.

In System Log (CEX2C without CEX2A Systems):

```

S CSF
CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.
CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2 COPROCESSOR Epp,
SERIAL NUMBER nnnnnnnn.
CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.
CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn,
ACTIVE.

```

Message CSFM434E will be issued for each CEX2C online.

Message CSFM441I will be issued for each active CEX2C.

In System Log (CEX2C and CEX2A Systems):

```

S CSF
CSFM101E PKA KEY DATA SET, CSF.PKDS IS NOT INITIALIZED.
CSFM434E INCORRECT MASTER KEY (SYM) ON CRYPTO EXPRESS2
COPROCESSOR Epp,SERIAL NUMBER nnnnnnnn.
CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE
CSFM100E CRYPTOGRAPHIC KEY DATA SET, CSF.CKDS IS NOT INITIALIZED.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
CSFM441I CRYPTO EXPRESS2 COPROCESSOR Epp, SERIAL NUMBER nnnnnnnn,
ACTIVE.

```

Message CSFM434E will be issued for each CEX2C online.

Message CSFM435I will be issued for each active CEX2A.

Message CSFM441I will be issued for each active CEX2C.

In System Log (CPACF only (no CEX2C or CEX2A) System):

```

S CSF
CSFM507I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC COPROCESSORS ONLINE.
CSFM508I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC ACCELERATORS ONLINE.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

```

In System Log (CPACF and CEX2A)

```

S CSF
CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE
CSFM507I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC COPROCESSORS ONLINE.
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.

```

Message CSFM435I will be issued for each active CEX2A.

References

z/OS Cryptographic Services 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, proceed to the next topic.

Process Follow the instructions in Appendix B, "CICS-ICSF Attachment Facility," on page 215 if desired.

Responsible System Programmer

Where	Sample Jobs
References	Appendix B, "CICS-ICSF Attachment Facility," on page 215
Completed	

Step 9. Complete ICSF initialization

See "Steps for initializing ICSF" on page 26

Responsible System Programmer

Where Operator Console

Completed

Commonly Encountered ICSF First Time Setup/initialization Messages

These 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 (CCFs) 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.
This message is only if you have a CCF System.
- **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.
This message is only if you have a CCF System.
- **CSFM410E ERROR IN OPTIONS DATA SET** - ICSF could not interpret the options data set.

CCF Systems ONLY: 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.

A 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 these 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.

Steps for setting up ICSF

Perform these 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 these 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 Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.
 - 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 Cryptographic Services 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 13.

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. z890, z990, z9 EC, z9 BC and System z10 Enterprise Class without a PCI XCC or CEX2CC

This topic describes the processing of the IBM @server zSeries 990, z890, z9 EC, z9 BC and System z10 Enterprise Class environment, without a PCI X Cryptographic Coprocessor or Crypto Express2 Coprocessor. Note that these servers do not support the Cryptographic Coprocessor Feature or the PCI Cryptographic Coprocessor.

Applications and programs

Applications requiring secure cryptography using encrypted keys will not be able to execute on the z890, z990, or z9 EC or z9 BC without a PCI X Cryptographic Coprocessor or Crypto Express2 Coprocessor. All cryptographic keys must be clear keys.

These applications and programs are not supported:

- Access Method Services Cryptographic option
- CICS attachment facility
- CKDS Conversion program
- CSFEUTIL program for CKDS reencipher, refresh, change master key, and passphrase initialization functions
- CSFPUTIL program for PKDS activate, cache refresh, reencipher, and initialization functions
- Distributed Key Management System (DKMS)
- Key Generation Utility Program (KGUP)
- PCF applications
- UDX (User Defined Extension) support
- VTAM Session Level Encryption
- 4753-HSP applications
- Applications that access ICSF services through the BSAFE interfaces
- If only the CPACF feature is installed you will not be able to:
 1. Set master keys (SYM-MK and ASYM-MK)
 2. Initialize the CKDS and PKDS
 3. Store keys in the CKDS and PKDS.

Callable services

These services are available when running on a z990, z890, or z9 EC or z9 BC without a PCI X Cryptographic Coprocessor or Crypto Express2 Coprocessor:

- Character/Nibble Conversion (CSNBXBC and CSNBXCB)
- Code Conversion (CSNBXEA and CSNBXAE)
- Control Vector Generate (CSNBCVG)
- Decode (CSNBDCO) - This service requires CP Assist for Cryptographic Functions.
- Digital Signature Verify (CSNDDSV) - This service requires a PCI Cryptographic Accelerator.
- Encode (CSNBECO) - This service requires CP Assist for Cryptographic Functions.

- ICSF Query Service (CSFIQF) - The only part of this service available without a PCIXCC/CEX2C is the ICSFSTAT function.
- MDC Generate (CSNBMDG and CSNBMDG1) - This service requires CP Assist for Cryptographic Functions.
- One-Way Hash Generate (CSNBOWH and CSNBOWH1) - This service requires CP Assist for Cryptographic Functions.
- 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)
- PKCS11 Token Record Create (CSFPTRC)
- PKCS11 Token Record Delete (CSFPTRD)
- PKCS11 Token Record List (CSFPTRL)
- PKCS11 Get Attribute Value (CSFPGAV)
- PKCS11 Set Attribute Value (CSFPSAV)
- Symmetric Key Decipher (CSNBSYD and CSNBSYD1) - This service requires CP Assist for Cryptographic Functions.
- Symmetric Key Encipher (CSNBSYE and CSNBSYE1) - This service requires 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 supported callable services.

ICSF Setup and Initialization

Starting ICSF on a z990, z890, or z9 EC or z9 BC without a PCI Cryptographic Accelerator or PCIXCC/CEX2C or starting ICSF on a z9 EC or z9 BC without a Crypto Express2 Accelerator or CEX2C:

- Starting ICSF on a z990, z890, z9 EC or z9 BC without a PCI Cryptographic Accelerator or PCIXCC/CEX2C:

```
S CSF
$HASP100 CSF ON STCINRDR
IEF695I START CSF WITH JOBNAME CSF IS ASSIGNED TO USER
+++++++
$HASP373 CSF STARTED
IEF403I CSF - STARTED - TIME=11.07.28
CSFM506I CRYPTOGRAPHY - THERE IS NO ACCESS TO ANY CRYPTOGRAPHIC COPROCESSORS OR
ACCELERATORS.
CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
```

- Starting ICSF on a z990 or z890 with a PCI Cryptographic Accelerator and without a PCI X Cryptographic Coprocessor or Crypto Express2 Coprocessor, you'll receive message CSFM411I for each PCI Cryptographic Accelerator that is Active. Starting ICSF on a z9 EC, z9 BC and System z10 Enterprise Class with a Crypto Express2 Accelerator, and without a Crypto Express2 Coprocessor, you'll receive message CSFM435I for each Crypto Express2 Accelerator that is Active.

```
S CSF
$HASP100 CSF ON STCINRDR
IEF695I START CSF WITH JOBNAME CSF IS ASSIGNED TO USER
+++++++
```

```
$HASP373 CSF STARTED
IEF403I CSF - STARTED - TIME=11.08.15
CSFM411I PCI CRYPTOGRAPHIC ACCELERATOR Ann is ACTIVE (for z890/z990)
CSFM435I CRYPTO EXPRESS2 ACCELERATOR Fpp IS ACTIVE (for z9 EC/z9 BC)
CSFM507I CRYPTOGRAPHY - THERE ARE NO CRYPTOGRAPHIC COPROCESSORS ONLINE.
CSFM122I PKA SERVICES WERE NOT ENABLED DURING ICSF INITIALIZATION
CSFM001I ICSF INITIALIZATION COMPLETE
CSFM400I CRYPTOGRAPHY - SERVICES ARE NOW AVAILABLE.
```

Secure Sockets Layer (SSL)

| System SSL applications are supported on the z990, z890, z9 EC, z9 BC and
| System z10 Enterprise Class. 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 have been enhanced for DES,
TDES and SHA-1 services.

TKE workstation

The Trusted Key Entry (TKE) workstation is not available with this hardware
configuration.

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 such products 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 Vol 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.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

<http://www.ibm.com/systems/z/os/zos/bkserv/>

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:

American Express	American Express Company
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

- 4753
 - key tokens 55
- 4753-HSP
 - compatibility and coexistence with ICSF 161

A

- abends 83
- access control checking
 - udx 135
- Access Method Services Cryptographic Option and ICSF 69
- accessibility 237
- activity report
 - defining on a DD statement 155
 - description 156
- addressing mode
 - no restrictions on ICSF's caller 68
- AMS DEFINE CLUSTER command 15, 17, 20
- AMS IMPORT/EXPORT commands 15, 17, 20
- AMS REPRO command 15, 17, 20
- AMS REPRO encryption 144

C

- callable services
 - ICSF 44
- CANCEL command 59
- canceled ICSF 59
- CCA Token Export Key Parameters TLV Object 191
- CDMF 7
- changing parameters in installation options data set
 - specifying option keywords and values 32
- changing the master key in compatibility or coexistence mode 145
- CHECKAUTH installation option 32
- choosing compatibility modes during migration 146
- CICS
 - WAITLIST installation option 41
- CICS wait list 53
- CICS-ICSF Attachment Facility 215
 - installing 215
- CIPHER macro
 - SVC description 82
- CKDS
 - create 15
 - primary space required 15
 - secondary space required 15
- CKDS (cryptographic key data set) 8
 - conversion from PCF CKDS to ICSF CKDS 147
 - creating 16
 - description 8
 - header record format 163
 - record format 164
- CKDS entry retrieval installation exit
 - environment 110

- CKDS entry retrieval installation exit (*continued*)
 - input 111
 - installing 110
 - purpose and use 110
 - return codes 112
- CKDS refresh
 - SMF record type 82 73
- CKDSN installation option 32
- CKTAUTH 33
- clear master key part entry
 - SMF record type 82 74
- coexistence mode
 - changing the master key 145
 - description 143, 144
- coexistence with 4753-HSP 161
- coexistence, definition 43
- Common Export Key Parameters TLV Object 189
- COMPAT installation option 33, 143
- compatibility mode
 - and the Access Method Services Cryptographic Option 69
 - changing the master key 144, 145
 - description 143, 144
- compatibility with 4753-HSP 161
- COMPENC installation option 33
- component trace 80
- configure on/off cryptographic coprocessors 65
- controlling access to the callable services 78
- controlling access to the cryptographic keys 78
- controlling access to the key generator utility
 - program 78
- controlling the program environment 77
- conversion considerations
 - 4753-HSP to OS/390 ICSF 55
- conversion program
 - activity report 156
 - bypassing entries 151
 - converting key types 152
 - data sets 155
 - including information in a key entry 152
 - installation exit 148
 - JCL for submitting 155
 - override file 149
 - running 154
- conversion program installation exit
 - PCF 112
 - purpose and use 112
 - return codes 114
- converting a PCF CKDS 147
- Coprocessor Management panel 65
- CP Assist for Cryptographic Functions
 - description 2
- creating the CKDS
 - allocating space for the CKDS 15
 - reading the CKDS into storage 26
 - using the AMS DEFINE CLUSTER command 15
- creating the installation options data set
 - guidelines 21

- creating the PKDS
 - allocating space for the PKDS 17
- creating the startup procedure 23
 - specifying the CSFLIST data set 24
 - specifying the installation options data set 23
- creating the TKDS
 - allocating space for the TKDS 19
- Crypto Express2 Coprocessor
 - description 1
- cryptographic communication vector table 200
- cryptographic communication vector table
 - extension 207
- Cryptographic Coprocessor clear master key entry
 - SMF record type 82 74
- Cryptographic Coprocessor Feature
 - description 2
- cryptographic coprocessor retained key create or delete
 - SMF record type 82 75
- cryptographic coprocessor TKE command request or reply
 - SMF record type 82 75
- cryptographic coprocessors
 - bringing offline 65
 - bringing online 65
 - disabling 65, 66
- csf 23
- CSFAPRPC processing routine 136
- CSFCKDS exit 109
- CSFCONVX exit 112
- CSFESECI exit 119
- CSFESECK exit 119
- CSFESECS exit 119
- CSFESECT exit 119
- CSFEXIT1 exit 92
- CSFEXIT2 exit 92
- CSFEXIT3 exit 92
- CSFEXIT4 exit 92
- CSFEXIT5 exit 93
- CSFKGUP exit 122
- CSFLIST data set 24
- CSFPARAM data set 24
- CSFPRM00 22
- CSFPRM01 22
- CSFSRRW exit 115
- CSFVINP data set 155
- CSFVNEW data set 155
- CSFVOVR data set 155
- CSFVRPT data set 156
- CSFVSRRC data set 155

D

- DEFINE CLUSTER command 15, 17, 20
- defining conversion program data sets 155
- DES
 - with PKA 3
- DES external key token format 168
- DES with PKA 2
- disability 237
- disabling cryptographic coprocessors 65, 66
- DOMAIN installation option 34

- DSS private external key token 176
- DSS private internal key token 182
- DSS public token 171
- dynamic CKDS update
 - SMF record type 82 73
- dynamic PKDS update
 - SMF record type 82 74

E

- EMK macro
 - SVC description 82
- error handling for ICRF
 - SMF record type 82 73
- event recording 70
- exit
 - CKDS entry retrieval installation exit 88, 109
 - description 87
 - entry and return specifications 89
 - identifier on ICSF 34
 - invocation on ICSF 35
 - key generator utility program installation exit 89, 122
 - mainline installation exits 87, 92
 - PCF conversion program installation exit 88, 112
 - security installation exits 118
 - service installation exits 88, 99
 - single-record, read-write installation exit 88, 115
- EXIT installation option 34
- exit name table 97
- Export Key Rule Reference TLV Object 190
- external key token
 - DES 168
 - PKA
 - DSS private 176
 - RSA private 172

F

- FMID
 - applicable z/OS releases 6
 - hardware 6
 - servers 6
- formatting control blocks
 - using IPCS 83

G

- GENKEY macro
 - SVC description 82

H

- hardware features
 - IBM @server zSeries 800 5
 - IBM @server zSeries 890 5
 - IBM @server zSeries 900 5
 - IBM @server zSeries 990 4

I

- IBM @server zSeries 800
 - hardware features 5
- IBM @server zSeries 890
 - hardware features 5
- IBM @server zSeries 900
 - hardware features 5
- IBM @server zSeries 990
 - functions not supported 53, 54
 - hardware features 4
 - without PCI X Cryptographic Coprocessor 233
- ICSF
 - dispatching priority 42, 68
 - V1R2 and 4753-HSP key label considerations 55
- ICSF initialization
 - SMF record type 82 72
- ICSF interface changes
 - callable services 44
- ICSF status change
 - SMF record type 82 72
- icsf sysplex group
 - SMF record type 82 76
- initializing ICSF
 - creating the CKDS 16
 - creating the PKDS 17
 - creating the TKDS 20
 - creation of 16, 17, 20
 - selecting ICSF startup options
 - creating the installation options data set 21
 - creating the startup procedure 23
 - starting ICSF 26
- installation option keyword 32
 - CHECKAUTH 32
 - CKDSN 32
 - CKTAUTH 33
 - COMPAT 33, 143
 - COMPENC 33
 - DOMAIN 34
 - EXIT 34
 - KEYAUTH 38
 - MAXLEN 38
 - PKDSCACHE 38
 - PKDSN 38
 - REASONCODES 38
 - SERVICE 39
 - SSM 39
 - SYSPLEXCKDS 40
 - SYSPLEXTKDS 40
 - TKDSN 41
 - TRACEENTRY 41
 - UDX 41
 - USERPARM 41
 - WAITLIST 41
- installation options
 - performance considerations 67
- installation options data set 13, 21
 - changing option keywords and values 32
 - creating 21
 - example 22
 - specifying the installation options data set 23
- installation steps 13

- installation-defined service
 - access control checking 135
 - defining 135
 - description 133
 - entry and exit code example 134
 - executing 136
 - link editing 135
 - parameter checking 135
 - writing 133
- internal key token
 - aes; 165
 - clear 167
 - DES 166
 - PKA
 - DSS private 182
 - RSA private 177, 179

K

- key generator utility program exit parameter block 124
- key generator utility program installation exit
 - calling points 122
 - environment 123
 - installing 124
 - processing 123
 - purpose and use 122
 - return codes 132
 - SET statement 132
- key labels
 - differences between ICSF/MVS Version 1 Release 2 and 4753-HSP 55
- key part entry
 - SMF record type 82 73
- key token
 - aes; internal 165
 - DES
 - external 168
 - null 169
 - DES internal 166, 167
 - PKA 170
 - DSS private external 176
 - DSS private internal 182
 - DSS public 171
 - RSA 1024-bit modulus-exponent private
 - external 173
 - RSA 1024-bit private internal 179
 - RSA 2048-bit Chinese remainder theorem private
 - internal 181
 - RSA 4096-bit Chinese remainder theorem private
 - external 174
 - RSA private external 172
 - RSA private internal 177
 - RSA public 171
 - Trusted block 184
- KEYAUTH installation option 38
- keyboard 237

L

- link editing
 - callable services 135

M

- mainline installation exit
 - environment 93
 - exit parameter block 94
 - input 94
 - installing 93
 - parameters 95, 99
 - purpose and use 92
- master key part entry
 - SMF record type 82 73
- MAXLEN installation option 38
- message recording 77
- migrating from PCF 143
- migration
 - terminology 43
- migration considerations
 - 4753-HSP to OS/390 ICSF 55
- MODIFY command 59
- modifying ICSF 59

N

- noncompatibility mode
 - description 143, 146
- Notices 239
- null key token
 - format 169

O

- override file
 - defining on a DD statement 155

P

- panels
 - accessing 24
 - CSF@PRIM — Primary Menu 64
 - CSFGCMP0 — Coprocessor Management 65
- parameter checking
 - callable services 135
- PCF
 - application 144, 146
 - macro 143
 - migration to ICSF 143
- PCF (Programmed Cryptographic Facility)
 - cannot be cancelled and restarted 59
- PCF conversion program installation exit
 - environment 112
 - input 113
 - installing 113
 - purpose and use 112
- PCI Cryptographic Accelerator
 - description 2
- PCI Cryptographic Coprocessor
 - description 3
- PCI Cryptographic Coprocessor configuration
 - SMF record type 82 75
- PCI Cryptographic Coprocessor timing
 - SMF record type 82 75

- PCI X Cryptographic Coprocessor
 - description 1
- PCI X Cryptographic Coprocessor timing
 - SMF record type 82 76
- performance
 - problems 42, 68
- PKA key part entry
 - SMF record type 82 74
- PKA key token 170
 - record format
 - DSS private external 176
 - DSS private internal 182
 - DSS public 171
 - RSA 1024-bit modulus-exponent private external 173
 - RSA 1024-bit private internal 179
 - RSA 2048-bit Chinese remainder theorem private internal 181
 - RSA 4096-bit Chinese remainder theorem private external 174
 - RSA private external 172
 - RSA private internal 177
 - RSA public 171
- PKA master keys 8
- PKA token
 - record format
 - Trusted block 184
- PKDS (public key data set) 9
 - creating 17
 - description 9
 - header record format 169
 - record format 170
- PKDSCACHE installation option 38
- PKDSN installation option 38
- PKSC commands
 - SMF record type 82 74
- private external key token
 - DSS 176
 - RSA 172
- private internal key token
 - DSS 182
 - RSA 177, 179
- Protection Information TLV Object 187
- public key data set 9
 - improving security and reliability for the PKDS 17
- public key token
 - DSS 171
 - RSA 171

R

- read-write exit parameter block 117
- REASONCODES installation option 38
- recording events 70
- RETKEY macro
 - SVC description 82
- return codes
 - from PCF macros
 - migration consideration 144
- RMF
 - header record format 212

- RSA 1024-bit private internal key token 179
- RSA private external Chinese remainder theorem key token 174
- RSA private external key token 172
- RSA private external modulus-exponent key token 173
- RSA private internal Chinese remainder theorem key token 181
- RSA private internal key token 177
- RSA public token 171
- Rule Subsections
 - Trusted Block Token 188
- running ICSF
 - in coexistence mode 144
 - in compatibility mode 144
 - in noncompatibility mode 146
- running the conversion program
 - creating a job to run the conversion program 154
 - defining conversion program data sets 155

S

- scheduling changes for cryptographic keys 79
- secondary parameter block 107
- security considerations 77
- security installation exit
 - environment 119
 - input 121
 - installing 120
 - purpose and use 118
 - return codes 122
- selecting ICSF startup options
 - creating the installation options data set 21
 - creating the startup procedure 23
- service installation exit
 - environment 100
 - exit parameter block 104
 - input 104
 - installing 100
 - parameters 109
 - purpose and use 100
 - return codes 109
- SERVICE installation option 39
 - syntax 135
- service stub
 - description 133
 - example 141
 - linking 136
 - writing 136
- SET Certificate Authority 3
- shortcut keys 237
- single-record, read-write installation exit
 - conversion program invocation 148
 - input 116
 - installing 116
 - purpose and use 115
 - return codes 118
- SMF record type 82 70
 - subtype 1 72
 - subtype 10 74
 - subtype 11 74
 - subtype 12 74
- SMF record type 82 *(continued)*
 - subtype 13 74
 - subtype 14 74
 - subtype 15 75
 - subtype 16 75
 - subtype 17 75
 - subtype 18 75
 - subtype 19 76
 - subtype 20 76
 - subtype 21 76
 - subtype 22 76
 - subtype 23 77
 - subtype 3 72
 - subtype 4 73
 - subtype 5 73
 - subtype 6 73
 - subtype 7 73
 - subtype 8 73
 - subtype 9 73
- SMF recording 70, 132
- special secure mode
 - SMF record type 82 73
- specifying the installation options data set 23
- SSM installation option 39
- START command 26, 59
- starting ICSF
 - creating the startup procedure 23
 - entering the ICSF START command 26, 58
- startup procedure 13, 23
- steps in installation 13
- STOP command 59
- stopping ICSF 58
- SVC 143 82
- SYS1.PARMLIB
 - customizing 14
 - description 13
- SYS1.PROCLIB
 - description 13
 - storing startup procedure 24
- SYS1.SAMPLIB
 - CSFPRM00 22
 - CSFPRM01 22
 - description 13
- SYSPLEXCKDS installation option 40
- SYSPLEXTKDS installation option 40

T

- testing ICSF 146
- TKDS
 - SMF record type 82 77
- TKDS (public key data set)
 - creating 20
- TKDS (token data set)
 - description 49
 - format 192
- TKDS (token key data set) 9
 - description 9
- TKDSN installation option 41
- token data set (TKDS)
 - description 49

- token data set (TKDS) (*continued*)
 - format 192
- token key data set 9
 - improving security and reliability for the TKDS 20
- token validation value (TVV) 167
- TRACEENTRY installation option 41
- Transport Key Rule Reference TLV Object 188
- Transport Key Variant TLV Object 188
- triple DES
 - for data privacy 3
- trusted block create
 - SMF record type 82 76
- trusted block token
 - PKA
 - trusted block 184
- Trusted block token 184

U

- udx
 - access control checking 135
- UDX installation option 41
- UDX support 55
- User Defined Extension 55
- USERPARM installation option 41
- using different configurations 59
- using the conversion program override file 149

V

- virtual storage constraint relief
 - for the caller of ICSF 68
- VSAM data set
 - creating 15
- VTAM
 - starting before ICSF 58
- VTAM session-level encryption
 - and ICSF 68

W

- WAITLIST installation option 41

Readers' Comments — We'd Like to Hear from You

z/OS
Cryptographic Services
Integrated Cryptographic Service Facility
System Programmer's Guide

Publication No. SA22-7520-12

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

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. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: mhvrfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



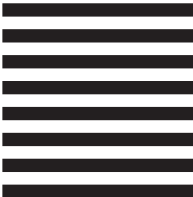
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
MHVRCFS, Mail Station P181
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

Printed in USA

SA22-7520-12

