z/OS

IBM

Cryptographic Services Integrated Cryptographic Service Facility Administrator's Guide

Version 2 Release 2

SC14-7506-06

Note

Before using this information and the product it supports, read the information in "Notices" on page 427.

This edition applies to ICSF FMID HCR77C0 and Version 2 Release 2 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this information

This information describes how to manage cryptographic keys by using the z/OS Integrated Cryptographic Service Facility (ICSF), which is part of z/OS Cryptographic Services. The z/OS Cryptographic Services include these components:

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

ICSF is a software product that works with the hardware cryptographic feature and the z/OS 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. The cryptographic coprocessor is secure, high-speed hardware that performs the actual cryptographic functions. The cryptographic feature available to your applications depends on the server or processor hardware.

References to the IBM zEnterprise 114 do not appear in this information. Be aware that the documented notes and restrictions for the IBM zEnterprise 196 also apply to the IBM zEnterprise 114.

ICSF Features

ICSF enhances z/OS security as follows:

- It ensures data privacy by encrypting and decrypting the data.
- It manages personal identification numbers (PINs).
- It ensures the integrity of data through the use of modification detection codes (MDCs), hash functions, or digital signatures.
- It ensures the privacy of cryptographic keys themselves by encrypting them under a master key or another key-encrypting key.
- It enforces AES and DES key separation, which ensures that cryptographic keys are used only for their intended purposes.
- It enhances system availability by providing continuous operation.
- It enables the use of Rivest-Shamir-Adelman (RSA) and Elliptic Curve Cryptography (ECC) public and private keys on a multi-user, multi-application platform.
- It provides the ability to generate RSA and ECC key pairs within the secure hardware boundary of the cryptographic hardware features.

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

This information explains the basic concepts of protecting and managing the keys used in cryptographic functions. It provides step-by-step guidance for the ICSF administration tasks.

Who should use this information

This information is intended for anyone who manages cryptographic keys. Usually, this person is the ICSF administrator.

The ICSF administrator performs these major tasks:

- Entering and changing master keys
- Generating, entering, and updating cryptographic keys
- Viewing system status, which includes hardware status, installation options, installation exits, and installation services

How to use this information

The first five topics give you background information you need to manage cryptographic keys on ICSF.

- Chapter 1, “Introduction,” on page 1 gives a brief introduction to the role of cryptography in data security. It describes the cryptographic algorithms that ICSF supports and discusses the importance of key secrecy.
- Chapter 2, “Understanding cryptographic keys,” on page 11 describes how ICSF protects keys and controls their use. It also describes the types of keys and how ICSF protects data and keys within a system and outside a system.
- Chapter 3, “Managing cryptographic keys,” on page 31 describes how to manage keys with ICSF. It introduces how to generate or enter, maintain, and distribute keys using ICSF. It also describes how to use keys to distribute keys and PINs between systems.
- Chapter 4, “Setting up and maintaining cryptographic key data sets,” on page 69 introduces the cryptographic key data sets for ICSF. It also describes metadata that can be used in managing keys and objects.
- Chapter 5, “Controlling who can use cryptographic keys and services,” on page 81 describes how you can control access to, and use of, cryptographic keys and services.

The remaining topics describe how to use the ICSF panels to manage cryptographic keys and also to view system status. Each topic gives background information about a major task and leads you through the panels, step-by-step, for the task.

- Chapter 6, “Using the pass phrase initialization utility,” on page 95 discusses pass phrase initialization and gives step-by-step instructions on how to get your cryptographic system up and running quickly. The pass phrase initialization utility allows you to install the necessary master keys on cryptographic coprocessors, and initialize the CKDS and PKDS with a minimal effort.
- Chapter 7, “Managing CCA Master Keys,” on page 103 describes how to enter, activate, and manage master keys with the CCA cryptographic coprocessors.
- Chapter 8, “Managing PKCS #11 master keys,” on page 159 describes how to manage master keys for the Enterprise PKCS #11 (EP11) coprocessors.
- Chapter 9, “Key management on systems without coprocessors,” on page 167, describes how to manage clear AES and DES DATA keys on a system that does not have any cryptographic coprocessors or accelerators.
- Chapter 10, “Running in a Sysplex Environment,” on page 171, describes various considerations when sharing the CKDS, PKDS, and TKDS in a sysplex environment.

- Chapter 11, “Managing Cryptographic Keys Using the Key Generator Utility Program,” on page 185, describes how to use the *key generator utility program* (KGUP). The program generates keys and stores them in the *cryptographic key data set* (CKDS).
- Chapter 12, “Viewing and Changing System Status,” on page 265, describes how to display information about parts of ICSF that your installation can specify and change. It describes how to use the panels to display installation options, hardware status, cryptographic processor status, installation exits, and installation-defined services.
- Chapter 13, “Managing User Defined Extensions,” on page 309, describes how to use panels to manage your own cryptographic callable service.
- Chapter 14, “Using the Utility Panels to Encode and Decode Data,” on page 313, describes how to use utility panels to encipher and decipher data with a key that is not enciphered.
- Chapter 15, “Using the utility panels to manage keys in the PKDS,” on page 317, describes how to use the PKDSKEYS option on the ICSF utilities panel to provide PKDS key management capability.
- Chapter 16, “Using PKCS11 Token Browser Utility Panels,” on page 325, describes how to use the PKCS11 TOKEN option on the ICSF utilities panel to provide TKDS key management capability.
- Chapter 17, “Using the ICSF Utility Program CSFEUTIL,” on page 343, describes how to use the CSFEUTIL utility program to change master keys and refresh or reencipher the CKDS.
- Chapter 18, “Using the ICSF Utility Program CSFPUTIL,” on page 353, describes how to use the CSFPUTIL utility program to reencipher and refresh a PKDS.
- Chapter 19, “Using the ICSF Utility Program CSFDUTIL,” on page 361 describes how to use the CSFDUTIL utility program to check the CKDS and PKDS for duplicate key tokens.
- Chapter 20, “Rewrapping DES key token values in the CKDS using the utility program CSFCNV2,” on page 365, describes how to use the CSFCNV2 utility to rewrap encrypted tokens in the CKDS.
- Chapter 21, “Using ICSF health checks,” on page 369, describes a set of health checks that inform you of potential ICSF problems.
- Appendix A. Appendix A, “ICSF Panels,” on page 387
- Appendix B, “Control Vector Table,” on page 395, contains a table of the control vector values that are associated with each key type.
- Appendix C, “Supporting Algorithms and Calculations,” on page 397, shows algorithms that are used to calculate checksums, verification patterns, and other values.
- Appendix D, “PR/SM Considerations during Key Entry,” on page 403, discusses additional considerations when running in PR/SM logical partition mode.
- Appendix E, “CCA access control points and ICSF utilities,” on page 407 describes the access control points for controling the utilities describe in this book.
- Appendix F, “Callable services affected by key store policy,” on page 409 lists the callable service parameters that are checked when key store policy is enabled.
- Appendix H, “Questionable (Weak) Keys,” on page 421, gives examples of questionable keys.

Where to find more information

The publications in the z/OS ICSF library include:

- *z/OS Cryptographic Services ICSF Overview*
- *z/OS Cryptographic Services ICSF Administrator's Guide*
- *z/OS Cryptographic Services ICSF System Programmer's Guide*
- *z/OS Cryptographic Services ICSF Application Programmer's Guide*
- *z/OS Cryptographic Services ICSF Messages*
- *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*
- *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*

These publications contain additional ICSF information:

- *z/OS MVS System Codes*

  This publication describes the 18F abend code ICSF issues.
- *z/OS MVS System Management Facilities (SMF)*

  This publication describes SMF record type 82, where ICSF records events.
- *z/OS MVS Initialization and Tuning Guide*
- *z/OS MVS Initialization and Tuning Reference*
- *z/OS MVS Programming: Callable Services for High-Level Languages*
- *z/OS MVS Programming: Authorized Assembler Services Guide*
- *z/OS MVS Programming: Extended Addressability Guide*
- *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*
- *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*
- *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*
- *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*
- *z/OS DFSMSdfp Storage Administration*
- *z/OS DFSMS Access Method Services Commands*
- The z/OS Information Centers, which may be selected from http://www-03.ibm.com/systems/z/os/zos/bkserv/.

Related Publications

- *S/390 Support Element Operations Guide, GC38-3108*
- *z/990 PR/SM Planning Guide, SB10-7036*
- *zSeries Hardware Configuration Manager User's Guide, SC33-7989*
- *zSeries Hardware Management Console Operations, SC28-6820*
- *IBM Security Architecture: Securing the Open Client/Server Distributed Enterprise, SC28-8135*
- *VTAM in a Parallel Sysplex Environment, SG24-2113*
- *RSA's Frequently Asked Questions About Today's Cryptography* , available on the World Wide Web. See RSA's home page at http://www.rsa.com.
- *Applied Cryptography, Second Edition, by Bruce Schneier*, by Bruce Schneier, John Wiley & Sons, Inc.

Information for the IBM PCIe Cryptographic Coprocessor is found on the web at http://www-03.ibm.com/security/cryptocards/pciecc/library.shtml. Information for the IBM 4764 PCI-X Cryptographic Coprocessor is found on the web at **http://www-03.ibm.com/security/cryptocards/pcixcc/library.shtml**.

- 4765 PCIe Cryptographic Coprocessor Installation Manual

- CCA Basic Services Reference and Guide for the IBM 4765 PCIe and IBM 4764 PCI-X Cryptographic Coprocessors
- Addendum to CCA Basic Services Reference and Guide for the IBM 4765 PCIe and IBM 4764 PCI-X Cryptographic Coprocessors

IBM Crypto Education

The IBM Crypto Education community provides detailed explanations and samples pertaining to IBM cryptographic technology at https://www.ibm.com/developerworks/community/groups/community/crypto.

How to send your comments to IBM

We appreciate your input on this documentation. Please provide us with any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

Use one of the following methods to send your comments:

Important: If your comment regards a technical problem, see instead "If you have a technical problem."

- Send an email to mhvrcfs@us.ibm.com.
- Send an email from the Contact z/OS.

Include the following information:

- Your name and address
- Your email address
- Your phone or fax number
- The publication title and order number:
  z/OS ICSF Administrator's Guide
  SC14-7506-06
- The topic and page number or URL of the specific information to which your comment relates
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one or more of the following actions:

- Visit the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

ICSF is an element of z/OS, but provides independent ICSF releases as web deliverables. These web deliverables are identified by their FMID. Each release of z/OS includes a particular ICSF FMID level as part of its base.

ICSF publications can be obtained from:
- IBM Resource Link at http://www.ibm.com/servers/resourcelink. (Click "publications" and then select the release that you are interested in under "ICSF Publications by FMID".)
- IBM z/OS downloads for Cryptographic Support at http://www.ibm.com/systems/z/os/zos/downloads.

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

Changes made in Cryptographic Support for z/OS V2R1 - z/OS V2R2 (FMID HCR77C0)

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SC14-7506-05.

This document is for ICSF FMID HCR77C0. This release of ICSF runs on z/OS z/OS V2R1 and z/OS V2R2 and only on zSeries hardware.

New

- Information about key fingerprints.
- "ICSF_OPTIONS_CHECKS" on page 376 health check.
- "ICSF_UNSUPPORTED_CCA_KEYS" on page 377 health check.

Changed

- "Setting up and maintaining the cryptographic key data set (CKDS)" on page 69 was updated.
- "Setting up and maintaining the public key data set (PKDS)" on page 72 was updated.
- "Variable-length metadata blocks" on page 78 was updated.
- "Setting up profiles in the CSFSERV general resource class" on page 83 was updated.
- "Enabling use of encrypted keys in callable services that exploit CPACF" on page 94 was updated.
- "Displaying installation options" on page 279 was updated.

Deleted

No content was removed from this information.

Changes made in Cryptographic Support for z/OS V1R13 - z/OS V2R2 (FMID HCR77B1) as updated April 2016

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SC14-7506-04.

This document is for ICSF FMID HCR77B1. This release of ICSF runs on z/OS V1R13, z/OS V2R1, and z/OS V2R2 and only on zSeries hardware.

New

- Updated to include information about IBM z13s.
- Added information about the Encrypted PIN Translate Enhanced and Key Encryption Translate services.
- Added information about regional cryptographic servers.

Changed

- Terminology changed from open cryptographic services to regional cryptographic services.

Deleted

No content was removed from this information.

Changes made in Cryptographic Support for z/OS V1R13 - z/OS V2R2 (FMID HCR77B1)

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SC14-7506-03.

This document is for ICSF FMID HCR77B1. This release of ICSF runs on z/OS V1R13, z/OS V2R1, and z/OS V2R2 and only on zSeries hardware.

New

- "Regional cryptographic servers" on page 5 is new.
- "Regional cryptographic operational keys" on page 20 is new.
- "Managing regional cryptographic keys" on page 67 is new.
- "Regional cryptographic server master key" on page 67 is new.
- CSFCMP00 - ICSF Coprocessor Management panel has been modified in "Displaying the EP11 domain roles" on page 300.
- "ICSF Primary Menu panel" on page 387 has been modified.
- "CSFCMP00 — Coprocessor Management panel" on page 390 has been modified.
- Appendix G, "Callable services that trigger reference date processing," on page 415 is new.

Changed

- "Setting up and maintaining the token data set (TKDS)" on page 74 was updated.
- "Setting up profiles in the CSFSERV general resource class" on page 83 was updated with information on the six new EMV services.
- Chapter 8, "Managing PKCS #11 master keys," on page 159 was updated.

- "Displaying administrative control functions" on page 266 was updated.
- "Displaying cryptographic coprocessor status" on page 267 was updated.
- "Changing coprocessor or accelerator status" on page 269 was updated.
- "Displaying coprocessor hardware status" on page 270 was updated.
- "Displaying installation options" on page 279 was updated.
- "Display CCA domain roles" on page 291 was updated.
- "Secret Key Object Details panel" on page 333 was updated.
- Appendix F, "Callable services affected by key store policy," on page 409 was updated with information on the six new EMV services.

Deleted

No content was removed from this information.

Changes made in Enhanced Cryptographic Support for z/OS V1R13 - z/OS V2R1 (FMID HCR77B0)

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SC14-7506-01.

This document is for ICSF FMID HCR77B0. This release of ICSF runs on z/OS V1R13 and z/OS V2R1 and only on zSeries hardware.

New

- Information added about "Crypto Express5 Feature (CEX5A, CEX5C or CEX5P)" on page 4.
- The following new information is added to "Controlling how cryptographic keys can be used" on page 51:
  - "Enabling use of archived KDS records" on page 61.
- The following new information is added to Chapter 4, "Setting up and maintaining cryptographic key data sets," on page 69:
  - "Key data set metadata" on page 75.
  - "Archiving and recalling a record in a key data set" on page 77
  - "Variable-length metadata blocks" on page 78
  - "IBM metadata blocks" on page 78
  - "Key material validity dates" on page 79
  - "Sharing KDS with older releases of ICSF and sysplex implications" on page 79
- The following new installation options have been added to the CSFSOP10 — Installation Options panel:
  - KEYARCHMSG (YES or NO)
  - RNGCACHE(YES or NO)

  See "Displaying installation options" on page 279 for additional details.
- New health checks are added:
  - "ICSF_DEPRECATED_SERV_WARNINGS" on page 372
  - "ICSF_KEY_EXPIRATION" on page 373
- The CKDS KEY CHECK option was added to the "CSFMKM20 — CKDS Management panel" on page 390 and the PKDS KEY CHECK option was added to the "CSFMKM30 — PKDS Management panel" on page 391.

Changed

- “Master keys” on page 11 has been updated.
- “Symmetric key separation” on page 21 has been updated.
- “Entering keys” on page 33 has been updated.
- “Entering master keys” on page 34 has been updated.
- “Entering keys into the CKDS” on page 35 has been updated.
- “Key Store Policy” on page 39 has been updated.
- Chapter 7, “Managing CCA Master Keys,” on page 103 has been updated.
- “DES master key” on page 107 has been updated.
- “Steps for entering the first master key part” on page 117 has been updated.
- “Initializing the key data sets at first-time startup” on page 128 has been updated.
- “Deactivating the last coprocessor” on page 270 has been updated.

Deleted

No content was removed from this information.

Changes made in Cryptographic Support for z/OS V1R13-V2R1 (FMID HCR77A1) as updated June 2014

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SC14-7506-00.

This document is for ICSF FMID HCR77A1. This release of ICSF runs on z/OS V1R13 and z/OS V2R1 and only on zSeries hardware.

New

- Support for the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods with the application of the PTFs for APARs OA42246, OA43906, and OA44444:
  - “Setting up profiles in the CSFSERV general resource class” on page 83 has been updated with new information.
  - Chapter 11, “Managing Cryptographic Keys Using the Key Generator Utility Program,” on page 185 has been updated with new information.
  - Appendix F, “Callable services affected by key store policy,” on page 409 has been updated with new information.
- The migration check “ICSFMIG_MASTER_KEY_CONSISTENCY” on page 380 was introduced in APAR OA39489 and has been added.

Changed

- “Using KGUP for key exchange” on page 187 has been updated.
- “Displaying installation options” on page 279 has been updated.
- Chapter 21, “Using ICSF health checks,” on page 369 has been updated.

Deleted

No content was removed from this information.

Changes made in Cryptographic Support for z/OS V1R13-V2R1 (FMID HCR77A1)

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SA22-7521-11, which supports z/OS Version 1 Release 13.

This document is for ICSF FMID HCR77A1. This release of ICSF runs on z/OS V1R13, and z/OS V2R1, and only on zSeries hardware.

New information

- A new section has been added to Chapter 7, "Managing CCA Master Keys," on page 103 describing "New master keys automatically set when ICSF started" on page 107
- The "Displaying installation options" on page 279 section of Chapter 12, "Viewing and Changing System Status," on page 265 has been updated with the new CTRACE option.
- A new "ICSFMIG77A1_COPROCESSOR_ACTIVE" on page 383 health check has been added to Chapter 21, "Using ICSF health checks," on page 369.
- A new section, "PKDS" on page 133, has been added to Chapter 7, "Managing CCA Master Keys," on page 103.
- "CSFCMP00 — Coprocessor Management panel" on page 390 has been modified.
- "System authorization facility (SAF) controls" on page 81 has been updated in multiple places to include information about:
  - Symmetric Key Export with Data, CSNDSXD
  - CSFOWH, CSFOWH1
  - CSFRNG CSFRNGL
- The Callable Service and its Exit Identifier table in "Displaying installation exits" on page 302 of Chapter 12, "Viewing and Changing System Status," on page 265 has been updated with new information about:
  - Authentication Parameter Generate
  - Recover PIN From Offset
  - Symmetric Key Export with Data
- Appendix F, "Callable services affected by key store policy," on page 409 has been updated with new information about:
  - Authentication Parameter Generate
  - Recover PIN From Offset
  - Symmetric Key Export with Data
  - Unique Key Derive
- The Operational Key Load, and the Operational Key Load - Variable-Length Tokens, access control points for callable services CSNBOKL / CSNEOKL have been added to the Appendix E, "CCA access control points and ICSF utilities," on page 407 appendix and removed from the ICSF Application Programmers's Guide.

Changed information

- "Displaying cryptographic coprocessor status" on page 267 in Chapter 12, "Viewing and Changing System Status," on page 265 has been updated with new and modified status values.

- Chapter 7, “Managing CCA Master Keys,” on page 103, section “CKDS” on page 132 has been updated with new and modified instructions.
- The following sections have been renamed, rewritten, enhanced, and reorganized for clarity:
  - Steps for performing a local change master key has been renamed “Changing the master keys” on page 140.
  - “Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls” on page 109 is new information within “Changing the master keys” on page 140.
  - “Symmetric master keys and the CKDS” on page 142
  - “Steps for reenciphering the CKDS and performing a local symmetric master key change” on page 143
  - “Asymmetric master keys and the PKDS” on page 145
  - Steps for enabling and disabling PKA callable services and PKDS updates, and Steps for performing a local PKDS change master key, have been replaced by “Steps for reenciphering the PKDS and performing a local asymmetric master key change” on page 146
- Many ICSF panel screen shots have been moved to appendix Appendix A, “ICSF Panels,” on page 387.

  All instructions previously displaying ICSF panels have been adjusted accordingly.

Deleted information

- In z/OS V2R1 support for z900/z800 and z990/z890 was removed. HCR77A1 removes support for the hardware present on z900/z800 (CCF and PCICC). This documentation reflects the removal of that support.

  z990/z890 (PCIXCC and PCICA) will continue to be supported because HCR77A1 runs on z/OS V1R13. In addition, a number of software functions (and their documentation) that rely on the presence of CCF and PCICC features or do not serve a purpose without these features is removed.
- The TRACEENTRY option has been deleted from the “Displaying installation options” on page 279 section of Chapter 12, “Viewing and Changing System Status,” on page 265.
- A caution was added to “Using KGUP for key exchange” on page 187 and “Syntax of the ADD and UPDATE control statements” on page 191 regarding KGUP not creating complementary key control statement or key label that has CLEAR parm specified in the KGUP statement.
- The section Sharing and migrating a CKDS and PKDS between a CCF system and a non-CCF system has been relocated to the ICSF Systems Programmer’s Guide.

Changes made in Cryptographic Support for z/OS V1R12-R13 (FMID HCR77A0)

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SA22-7521-16, which supports z/OS Version 1 Release 13.

This document is for ICSF FMID HCR77A0. This release of ICSF runs on z/OS V1R12, and z/OS V1R13, and only on zSeries hardware.

New information

- Coordinated change master key for PKDS and TKDS.
- Coordinated PKDS refresh.
- PKCS #11 master key management.
- KGUP has new key types and control statement keywords.
- New support for IBM zEnterprise EC12
- A new Appendix E, "CCA access control points and ICSF utilities," on page 407 appendix was added.
- Added new information to "Key strength and wrapping of key" on page 23.
- Added new information for Unique Key Derive to "Setting up profiles in the CSFSERV general resource class" on page 83 in "System authorization facility (SAF) controls" on page 81, Appendix B, "Control Vector Table," on page 395 and Appendix F, "Callable services affected by key store policy," on page 409.

Changed information

- Extensive changes were made to "Syntax of the ADD and UPDATE control statements" on page 191.
- "Using the ADD and UPDATE control statements for key management and distribution functions" on page 208 was updated for AES keys.
- Example control statements for "Syntax of the OPKYLOAD Control Statement" on page 216 have been updated.
- Cryptographic Key Data Set (CKDS) in "Specifying KGUP data sets" on page 224 has multiple updates.
- Chapter 6, "Using the pass phrase initialization utility," on page 95 was updated with new information about DES operational keys and the 24-byte DES master keys.
- "Algorithm for calculating a verification pattern" on page 399 was updated with new information about DES operational keys and the 24-byte DES master keys.
- "Pass Phrase Initialization master key calculations" on page 399 was updated with additional information about RSA and DES master keys.
- "DES master key" on page 107 in Chapter 7, "Managing CCA Master Keys," on page 103was updated with new information.

Chapter 1. Introduction

In today's business environment, data is one of the most valuable resources that is required for maintaining a competitive edge. As a result, businesses must often be able to maintain data secrecy, readily determine the authenticity of data, and closely control access to data.

Data systems commonly consist of many types and sizes of computer systems that are interconnected through many different electronic data networks. It is now common for an organization to interconnect its data systems with systems that belong to customers, vendors, and competitors. Larger organizations might include international operations, or they might provide continual services. As the Internet becomes the basis for electronic commerce and as more businesses automate their data processing operations, the potential for disclosing sensitive data to unauthorized persons increases. As a result, approaches to data security must provide:

- Common services for each computing environment
- Support for national and international standards
- Graduated degrees of support
- Flexibility to work with existing and emerging systems
- Management of the increased risks to data assets

A combination of elements must work together to achieve a more secure environment. To provide a foundation for a secure environment, a security policy should be based on the following:

- An appraisal of the value of data
- An analysis of the potential threats to that data

The Tasks of a Data Security System

To help you select the products and services that you need to put a data security policy into effect, IBM has categorized these security functions. These functions are based on the International Organization for Standardization (ISO) standard 7498-2:

- **Identification and authentication**—identifies users to the system and provides proof that they are who they claim to be.
- **Access control**—determines which users can access which resources.
- **Data confidentiality**—protects an organization's sensitive data from being disclosed to unauthorized individuals.
- **Data integrity**—ensures that data is in its original and unaltered form.
- **Security management**—administers, controls, and reviews a business security policy.
- **Nonrepudiation**—assures that a message sender cannot deny later that he or she sent the message.

The z/OS Integrated Cryptographic Service Facility (ICSF) provides a cryptographic application programming interface that you can use along with your system's cryptographic feature to put these functions into effect in your data security policy.

The Role of Cryptography in Data Security

Cryptography includes a set of techniques for scrambling or disguising data so that it is available only to someone who can restore the data to its original form. In current computer systems, cryptography provides a strong, economical basis for keeping data secret and for verifying data integrity.

ICSF supports these two main types of cryptographic processes:

- Symmetric algorithms, in which the same key value is used in both the encryption and decryption calculations
- Asymmetric algorithms, in which a different key is used in the decryption calculation than was used in the encryption calculation

Symmetric Cryptography

ICSF supports the following symmetric cryptography algorithms: The Data Encryption Algorithm and the Advanced Encryption Standard.

The Data Encryption Algorithm and the Data Encryption Standard

For commercial business applications, the cryptographic process that is known as the Data Encryption Algorithm (DEA)[1] has been widely adopted. The Data Encryption Standard (DES), as well as other documents, defines how to use the DES algorithm to encipher data. The Data Encryption Standard is the basis for many other processes for concealing data, such as protection of passwords and personal identification numbers (PINs). DES uses a key to vary the way that the algorithm processes the data. DES data-encrypting keys can be single-, double-, or triple-length. A single-length DES key is a 56-bit piece of data that is normally retained in 8 bytes of data. Each eighth bit of the key data is designated as a parity bit. A symmetric cryptographic system uses the same key both to transform the original data (plaintext) to its disguised, enciphered form (ciphertext) and to return it to its plaintext form.

The DES algorithm, which has been proven to be efficient and strong, is widely known. For this reason, data security is dependent on maintaining the secrecy of the cryptographic keys. Because the DES algorithm is common knowledge, you must keep the key secret to ensure that the data remains secret. Otherwise, someone who has the key that you used to encipher the data would be able to decipher the data. Key management refers to the procedures that are used to keep keys secret.

When you want someone to be able to confirm the integrity of your data, you can use the DES algorithm to compute a message authentication code (MAC). When used in this way, the DES algorithm is a powerful tool. It is almost impossible to meaningfully change the data and still have it produce the same MAC for a given key. The standardized approaches authenticate data such as financial transactions, passwords, and computer programs.

The originator of the data sends the computed MAC with the data. To authenticate the data, the receiver uses the DES algorithm to recompute the MAC. The receiver's application then compares this result with the MAC that was sent with the data. Someone could, of course, change both the data and the MAC. Therefore,

1. The Data Encryption Algorithm is often referred to as the DEA, the DES algorithm or just as DES. This information uses the term DES to refer to this algorithm.

the key that is used to compute the MAC must be kept a secret between the MAC's originator and the MAC's authenticator.

An alternative approach to data-integrity checking uses a standard key value and multiple iterations of the DES algorithm to generate a modification detection code (MDC). In this approach to data-integrity checking, the MDC must be received from a trusted source. The person who wants to authenticate the data recomputes the MDC and compares the result with the MDC that was sent with the data.

Advanced Encryption Standard

ICSF supports the Advanced Encryption Standard algorithm for data privacy. This provides strong encryption. Key lengths of 128-bits, 192-bits and 256-bits are supported. Secure key AES is available if running on an IBM System z9 EC, z9 BC, or later with the Nov. 2008 or later licensed internal code (LIC).

Asymmetric Algorithm or Public Key Cryptography

In an asymmetric cryptographic process one key is used to encipher the data, and a different but corresponding key is used to decipher the data. A system that uses this type of process is known as a public key system. The key that is used to encipher the data is widely known, but the corresponding key for deciphering the data is a secret. For example, many people can use your public key to send enciphered data to you with confidence, knowing that only you should possess the secret key for deciphering the data.

Public key cryptographic algorithms are used in processes that simplify the distribution of secret keys, assuring data integrity and provide nonrepudiation through the use of digital signatures.

The widely known and tested public key algorithms use a relatively large key. The resulting computer processing time makes them less than ideal for data encryption that requires a high transaction rate. Public key systems, therefore, are often restricted to situations in which the characteristics of the public key algorithms have special value, such as digital signatures or key distribution. PKA calculation rates are fast enough to enable the common use of digital signatures.

ICSF supports these public key algorithms:
- Rivest-Shamir-Adelman (RSA)
- Elliptic Curve Digital Signature Algorithm (ECDSA)

The RSA Public Key Algorithm

The Rivest-Shamir-Adelman (RSA)[2] public key algorithm is based on the difficulty of the factorization problem. The factorization problem is to find all prime numbers of a given number, n. When n is sufficiently large and is the product of a few large prime numbers, this problem is believed to be difficult to solve. For RSA, n is typically at least 512 bits, and n is the product of two large prime numbers. The ISO 9796 standard and provide more information about the RSA public key algorithm.

Elliptic Curve Digital Signature Algorithm (ECDSA)

The ECDSA algorithm uses elliptic curve cryptography (an encryption system based on the properties of elliptic curves) to provide a variant of the Digital Signature Algorithm.

2. Invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adelman

Cryptographic Hardware Features supported by z/OS ICSF

The cryptographic hardware available to your applications depends on your processor or server model. z/OS ICSF supports this hardware:

Crypto Express5 Feature (CEX5A, CEX5C or CEX5P)

- available on IBM z13 and z13s.
- contains one cryptographic engine that can be configured as a CCA cryptographic coprocessor (CEX5C), as an accelerator (CEX5A), or as a PKCS #11 cryptographic coprocessor (CEX5P).

Crypto Express4 Feature (CEX4A, CEX4C or CEX4P)

- available on IBM zEnterprise BC12 and EC12.
- contains one cryptographic engine that can be configured as a CCA coprocessor (CEX4C), PKCS #11 coprocessor (CEX4P) or as an accelerator (CEX4A).

Crypto Express3 Feature (CEX3C or CEX3A)

- available on IBM System z10 Enterprise Class, IBM System z10 Business Class, IBM zEnterprise 196, IBM zEnterprise 114, IBM zEnterprise EC12 and IBM zEnterprise BC12.
- contains two cryptographic engines that can be independently configured as a coprocessor (CEX3C) or as an accelerator (CEX3A)

Crypto Express2 Feature (CEX2C or CEX2A)

- available on IBM System z9 Enterprise Class, IBM System z9 Business Class, IBM System z10 Enterprise Class and IBM System z10 Business Class
- contains two cryptographic engines that can be independently configured as a coprocessor (CEX2C) or as an accelerator (CEX2A)
- provides support for clear keys in the CSNDDSV, CSNDPKD, and CSNDPKE callable services for better performance
- enables maximum SSL performance

CP Assist for Cryptographic Functions (CPACF)

CPACF is a set of cryptographic instructions providing improved performance. The servers support different algorithms:

- on the IBM System z9 Enterprise Class and IBM System z9 Business Class
  - SHA-1 algorithm is available
  - SHA-224 and SHA-256 algorithms are available
  - AES algorithm using 128-bit length keys is available
- on IBM System z10 Enterprise Class, z10 Business Class, IBM zEnterprise 196, IBM zEnterprise 114 and later
  - SHA-1 algorithm is available
  - SHA-224, SHA-256, SHA-384 and SHA-512 algorithms are available
  - AES algorithm using 128-, 192-, and 256-bit keys is available

CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement

CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement, feature 3863, provides for clear key DES and TDES instructions.

If you want to include cryptographic hardware, then feature 3863 is required.

If the CPACF feature is installed without the cryptographic hardware, you will not be able to:

1. Set master keys.
2. Initialize the PKDS or CKDS.
3. Store keys in the PKDS.

Regional cryptographic servers

Regional cryptographic servers are network-attached, standalone devices or dedicated Linux LPARs that perform geography-specific cryptography. These servers are secure key hardware security modules (HSMs) that operate similar to IBM's PKCS #11 secure coprocessors (CEX*n*P). They are marketed and serviced by third party vendors. Currently, the only geography-specific cryptography supported by these devices is the Chinese SM*x* family of algorithms. Secure keys are stored in the TKDS, protected by the Regional Cryptographic Server Master Key (RCS-MK).

The network-attached, stand-alone devices require no particular zSeries hardware, but do require communicating with z/OS V1R13 or later and ICSF FMID HCR77B1 or later. ICSF communicates with these devices using TCP/IP, with optional TLS protection. The Linux LPARs require IBM z13 or later hardware. ICSF communicates with the Linux LPARs using TCP/IP, with TLS protection required.

Once configured and online, ICSF makes the algorithms offered by these devices available as PKCS #11 vendor-defined extensions. For information on configuring these devices, see 'Adding and removing regional cryptographic servers' in *z/OS Cryptographic Services ICSF System Programmer's Guide*. For information on the algorithms offered, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* and *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Identification of cryptographic features

Starting in ICSF FMID HCR77B0, the prefix used to identify Crypto Express2, Crypto Express3, and Crypto Express4 features has changed. Table 1 lists the prefix for these features for FMIDs prior to HCR77B0 and the prefix for these features for FMID HCR77B0 and later releases. This change applies to ICSF messages, panels, and publications. The TKE workstation uses this same identification starting with TKE release 8.0.

Table 1. Cryptographic feature identification

Cryptographic feature	**Prefix for FMIDs prior to HCR77B0**	**Prefix for FMID HCR77B0 and later**
Crypto Express2 coprocessor	E	2C
Crypto Express2 accelerator	F	2A
Crypto Express3 coprocessor	G	3C
Crypto Express3 accelerator	H	3A
Crypto Express4 CCA coprocessor	SC	4C
Crypto Express4 EP11 coprocessor	SP	4P
Crypto Express4 accelerator	SA	4A

Note: All newer cryptographic features use the convention where *n* is the number in the feature name. For regional cryptographic servers, *n* represents the generation number of the server:

*n***A**	Crypto Express*n* accelerators.
*n***C**	Crypto Express*n* CCA coprocessors.
*n***P**	Crypto Express*n* EP11 coprocessors.
*n***R**	Regional cryptographic servers. Note: 2R requires ICSF FMID HCR77B1 with PTF OA49069 or later.

Managing Crypto Express2 Features on an IBM System z9 EC, z9 BC, z10 EC, and z10 BC

The Crypto Express2 feature can be configured as a coprocessor for secure key operation or as an accelerator for clear key RSA operations. The ability to change the configuration of the Crypto Express2 feature allows the administrator to change the Crypto Express2 to meet the site's processing needs. If master keys have been loaded into the registers on the Crypo Express2 feature, the master keys will be not be zeroized when the configuration is changed.

The Crypto Express2 is configured from the support element. See *Support Element Operations Guide*, SC28-6820, for details.

When changing the configuration:

- The coprocessor/accelerator must be deactivated on all partitions using that coprocessor/accelerator. From a z/OS System, you can do this using the ICSF coprocessor management panel or by issuing the SETICSF DEACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. This allows any existing work queued to the coprocessor/accelerator to complete and prevents new work from being enqueued.
- When the configuration change is complete (please allow sufficient time for the support element to complete the change), the coprocessor/accelerator can be activated on the ICSF coprocessor management panel or by issuing the SETICSF ACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. If the support element has not completed the change when a coprocessor/accelerator is activated, the status will be 'busy'.
- Coprocessors with valid master keys will become active and will be used to process work. Coprocessors without valid master keys will need to have a master key loaded. Accelerators will become active and will be used to process work.

Managing Crypto Express3 Features on an IBM System z10 EC, z10 BC, IBM zEnterprise 196, 114, BC12 and EC12

The Crypto Express3 feature can be configured as a coprocessor for secure key operation or as an accelerator for clear key RSA operations. The ability to change the configuration of the Crypto Express3 feature allows the administrator to change the Crypto Express3 to meet the site's processing needs. If master keys have been loaded into the registers on the Crypo Express3 feature, the master keys will be not be zeroized when the configuration is changed.

The Crypto Express3 is configured from the support element. See *Support Element Operations Guide*, SC28-6820, for details.

When changing the configuration:

- The coprocessor/accelerator must be deactivated on all partitions using that coprocessor/accelerator. From a z/OS System, you can do this using the ICSF coprocessor management panel or by issuing the SETICSF DEACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. This allows any existing work queued to the coprocessor/accelerator to complete and prevents new work from being enqueued.
- When the configuration change is complete (please allow sufficient time for the support element to complete the change), the coprocessor/accelerator can be activated on the ICSF coprocessor management panel or by issuing the SETICSF ACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. If the support element has not completed the change when a coprocessor/accelerator is activated, the status will be 'busy'.
- Coprocessors with valid master keys will become active and will be used to process work. Coprocessors without valid master keys will need to have a master key loaded. Accelerators will become active and will be used to process work.

Managing Crypto Express4 Features on an IBM zEnterprise BC12, EC12, z13, and z13s

The Crypto Express4 feature can be configured as a coprocessor for secure key operation or as an accelerator for clear key RSA operations. If configured as a coprocessor, it may be configured for CCA or PKCS #11. When configured as the latter, it is known as an Enterprise PKCS #11 coprocessor.

The Crypto Express4 feature's configuration may be switched from a CCA coprocessor to an accelerator and back without undergoing zeroization. If master keys have been loaded into the registers on the Crypto Express4 feature, the master keys will not be zeroized when the configuration is changed.

Note: This is not true for the Enterprise PKCS #11 coprocessor configuration. A switch from CCA or accelerator to PKCS #11 will result in the zeroization of the CCA master keys (DES, AES, RSA, and ECC) and settings. A switch from PKCS #11 to CCA or accelerator will result in the zeroization of the P11 master key and settings.

The Crypto Express4 is configured from the support element. See Support Element Operations Guide, SC28-6820, for details.

When changing the configuration:

- The coprocessor/accelerator must be deactivated on all partitions using that coprocessor/accelerator. From a z/OS System, you can do this using the ICSF coprocessor management panel or by issuing the SETICSF DEACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. This allows any existing work queued to the coprocessor/accelerator to complete and prevents new work from being enqueued.
- When the configuration change is complete (please allow sufficient time for the support element to complete the change), the coprocessor/accelerator can be activated on the ICSF coprocessor management panel or by issuing the SETICSF ACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. If the support element has not completed the change when a coprocessor/accelerator is activated, the status will be 'busy'.

- Coprocessors with valid master keys will become active and will be used to process work. Coprocessors without valid master keys will need to have a master key loaded. Accelerators will become active and will be used to process work.

Managing Crypto Express5 Features on an IBM z13 and z13s

The Crypto Express5 feature can be configured as a coprocessor for secure key operation or as an accelerator for clear key RSA operations. If configured as a coprocessor, it may be configured for CCA or PKCS #11. When configured as the latter, it is known as an Enterprise PKCS #11 coprocessor.

The Crypto Express5 feature's configuration may be switched from a CCA coprocessor to an accelerator and back without undergoing zeroization. If master keys have been loaded into the registers on the Crypto Express5 feature, the master keys will not be zeroized when the configuration is changed.

Note: This is not true for the Enterprise PKCS #11 coprocessor configuration. A switch from CCA or accelerator to PKCS #11 will result in the zeroization of the CCA master keys (DES, AES, RSA, and ECC) and settings. A switch from PKCS #11 to CCA or accelerator will result in the zeroization of the P11 master key and settings.

The Crypto Express5 is configured from the support element. See *Support Element Operations Guide*, SC28-6820, for details.

When changing the configuration:
- The coprocessor/accelerator must be deactivated on all partitions using that coprocessor/accelerator. From a z/OS System, you can do this using the ICSF coprocessor management panel or by issuing the SETICSF DEACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. This allows any existing work queued to the coprocessor/accelerator to complete and prevents new work from being enqueued.
- When the configuration change is complete (please allow sufficient time for the support element to complete the change), the coprocessor/accelerator can be activated on the ICSF coprocessor management panel or by issuing the SETICSF ACTIVATE console command if your system is running ICSF FMID HCR77B1 or later. If the support element has not completed the change when a coprocessor/accelerator is activated, the status will be 'busy'.
- Coprocessors with valid master keys will become active and will be used to process work. Coprocessors without valid master keys will need to have a master key loaded. Accelerators will become active and will be used to process work.

Strength of Hardware Cryptography

Cryptographic algorithms can be implemented in both software and specialized hardware. A hardware solution is often desirable because it provides these advantages:
- More secure protection to maintain the secrecy of keys
- Greater transaction rates

If a data security threat comes from an external source, a software implementation of the cryptographic algorithm might be sufficient. Unfortunately, however, much fraud originates with individuals within the organization (insiders). As a result, specialized cryptographic hardware can be required to protect against both insider and outsider data security threats. Well-designed hardware can:

- Ensure the security of cryptographic keys
- Ensure the integrity of the cryptographic processes
- Limit the key-management activities to a well-defined and carefully controllable set of services

The Role of Key Secrecy in Data Security

In both the symmetric key and asymmetric key algorithms, no practical means exists to identically cipher data without knowing the cryptographic key. Therefore, it is essential to keep a key secret at a cryptographic node. In real systems, however, this often does not provide sufficient protection. If adversaries have access to the cryptographic process and to certain protected keys, they could possibly misuse the keys and eventually compromise your system. A carefully devised set of processes must be in place to protect and distribute cryptographic keys in a secure manner.

ICSF, and other products that comply with the IBM Common Cryptographic Architecture (CCA), provide a means of controlling the use of cryptographic keys. This protects against the misuse of the cryptographic system.

This publication explains the concepts of key management and gives step-by-step instructions for using ICSF to generate, enter, and manage cryptographic keys.

Chapter 2. Understanding cryptographic keys

To understand cryptographic keys, you need to know the types of keys that exist and how ICSF protects them and controls their use. The Integrated Cryptographic Service Facility uses a hierarchical key management approach. A master key protects all the keys that are active on your system. Other types of keys protect keys that are transported out of the system. This topic gives you an understanding of how ICSF organizes and protects keys.

Values of keys

Keys can either be clear or encrypted. A clear key is the base value of a key. A clear key is not encrypted under another key. To create an encrypted key, either a master key or a transport key is used to encrypt the base value of the key.

Clear keys, if used carelessly, can compromise security. In symmetric cryptographic processes, such as DES or AES, anyone can use the clear key and the publicly known algorithm to decipher data, key values, or PINs. In asymmetric cryptographic processes it is important to protect the clear value of the private key. It would cause a serious security exposure if the wrong person obtained the value of the private key. It could be used to forge electronic signatures on documents, or decipher key values encrypted under the corresponding public key.

ICSF uses clear key values to *encode* and *decode* data. You can use the CCA callable services Symmetric Key Encipher and Symmetric Key Decipher to encode or decode data. You can use the Encode and Decode callable services or the ICSF utility panels to encode and decode data. For a description of the callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*. For a description of how to use the utility panels, see Chapter 14, "Using the Utility Panels to Encode and Decode Data," on page 313.

ICSF may have to input and output clear keys. For example, it might receive and send clear keys when it communicates with other cryptographic systems that use clear keys in their functions. When you give ICSF a clear key value, ICSF can encrypt the key before using it on the system. ICSF has specific callable services that perform this function. These callable services are clear key import and secure key import, which are described in *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Types of keys

ICSF groups the cryptographic keys into these categories:

- Master keys: CCA and PKCS #11.
- Operational keys: CCA, PKCS #11, and regional cryptographic server.

Master keys

ICSF uses master keys to protect other keys. Keys are active on a system only when they are encrypted under a master key variant, so the master key protects all keys that are used on the system. A key is in operational form when it has been encrypted under a master key variant. A key must be in operational form to be used with the cryptographic features.

The ICSF administrator initializes and changes master keys using the ICSF panels or TKE workstation. Master keys always remain in a secure area in the cryptographic hardware.

Master keys require cryptographic coprocessors. PKCS #11 or CCA coprocessors must be installed for operations using encrypted keys.

DES Master Key
The DES (DES-MK) master key is a 16-byte (128-bit) key that is used to protect symmetric DES/TDES keys used on all CCA coprocessors. The DES master key can be a 128-bit or 192-bit key on the zBC12, zEC12, and later systems with CEX3C or later coprocessor with the September 2012 or later licensed internal code.

AES Master Key
The AES (AES-MK) master key is a 32-byte (256 bit) key that is used to protect AES keys and HMAC keys on all CCA coprocessors. It is available on the z9 EC, z9 BC, and later servers with CEX2 or later coprocessors with the Nov. 2008 or later licensed internal code.

RSA Master Key
The RSA (RSA-MK) master key is a 24-byte (192-bit) key that is used to protect RSA private keys on all CCA coprocessors.

ECC Master Key
The ECC (ECC-MK) master key is a 32-byte (256 bit) key that is used to protect ECC keys and some RSA keys on CCA coprocessors. It is available on the z196, z114, and later systems with CEX3C or later coprocessor with the Sept. 2010 or later licensed internal code.

PKCS #11 Master Key
The PKCS #11 (P11-MK) master key is a 32-byte (256 bit) key that is used to protect secure PKCS #11 operational keys used on the Enterprise PKCS #11 coprocessor. It is available on the zEC12, zBC12, and later systems with CEX4P or later PKCS #11 coprocessors. For more information on PKCS #11 operational keys, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

CCA operational keys

ICSF supports symmetric and asymmetric cryptography and supports a variety of keys and functions.

Support of a key type or callable service is dependent on your hardware. See *z/OS Cryptographic Services ICSF Application Programmer's Guide* for details of hardware support.

Symmetric keys

ICSF supports AES, DES, and HMAC keys. ICSF groups the symmetric keys into these classes, which correspond to the functions they perform.

For a pair of complementary keys, such as the PIN generation and PIN verification key pair, the keys have the same clear key value. However, each key is protected by the control vector or associated data for its key type and the encrypted key values are different.

Data-encrypting keys: Data-encrypting keys are used to encrypt and decrypt data.

DATA class keys can be either encrypted under the master key or in the clear. CIPHER class are encrypted under the master key.

Table 2. DES data-encrypting keys

DES keys	Callable services
DATA class (data operation keys): • These key are used to encrypt and decrypt data. • Single-length keys can be used to generate and verify MACs and CVVs. • DATA keys can be single-length, double-length, or triple-length. • DATAM and DATAMV keys are double-length. • The DATA key value may be encrypted or clear. All other keys are encrypted.	
DATA (encrypted)	Authentication Parameter Generate, Cipher Text Translate2, CVV Key Combine, Decipher, Encipher, MAC Generate, MAC Verify, VISA CVV Generate, VISA CVV Verify
DATA (encrypted or clear)	Symmetric Key Encipher, Symmetric Key Decipher
DATAM	MAC Generate, MAC Verify
DATAMV	MAC Verify
Cipher class (data operation keys): • These key are used to encrypt and decrypt data. • The keys can be single-length or double-length.	
CIPHER	Cipher Text Translate2, Decipher, Encipher
DECIPHER	Cipher Text Translate2, Decipher
ENCIPHER	Cipher Text Translate2, Encipher

Table 3. AES data-encrypting keys

AES keys	Callable services
DATA class (data operation keys): • These key are used to encrypt and decrypt data. • Clear keys can be used to generate and verify MACs. • The keys can be 128, 192, or 256 bits in length. • The key value may be encrypted or clear.	
DATA (encrypted)	Cipher Text Translate2, Symmetric Algorithm Decipher, Symmetric Algorithm Encipher, Symmetric Key Decipher, Symmetric Key Encipher
DATA (clear)	Symmetric Key Decipher, Symmetric Key Encipher, Symmetric MAC Generate, Symmetric MAC Verify
Cipher class (data operation keys): • These key are used to encrypt and decrypt data. • The keys can be 128, 192, or 256 bits in length. • The key usage flags in the associated data can be used to restrict usage to encipher only or decipher only.	
CIPHER	Cipher Text Translate2, Symmetric Algorithm Decipher, Symmetric Algorithm Encipher

Availability notes: AES Cipher class keys require z114, z196, or later systems with a CEX3C or later coprocessor with the September 2011 or later licensed internal code.

Cipher text translation keys: Cipher text translation keys protect data that is transmitted through intermediate systems when the originator and receiver do not share a common key. Data that is enciphered under one cipher text translation key is reenciphered under another cipher text translation key on the intermediate node. During this process, the data never appears in the clear.

A cipher text translation key cannot be used in the decipher callable service to decipher data directly. It can translate the data from encipherment under one cipher text translation key to encipherment under another cipher text translation key. See "Protection of data" on page 29 for a description of how cipher text translation keys protect data that is sent through intermediate systems.

Table 4. DES cipher text translate keys

DES keys	Callable services
CIPHERXL class (cipher text translate keys): • These key are used to translate cipher text. • The keys are double-length.	
CIPHERXI	Cipher Text Translate2 (translate inbound key only)
CIPHERXL	Cipher Text Translate2 (translate inbound and outbound key)
CIPHERXO	Cipher Text Translate2 (translate outbound key only)

Availability notes: DES CIPHERXL class keys require zEC12, zBC12, and later systems with a CEX3C or later coprocessor with September 2012 or later licensed internal code.

Table 5. AES cipher text translate keys

AES keys	Callable services
CIPHERXL class (data operation keys): • These key are used to encrypt and decrypt data. • The keys can be 128, 192, or 256 bits in length. • The key usage flags in the associated data can be used to restrict usage to encipher only or decipher only. • The key usage flags in the associated data can be used to restrict usage to translate cipher text only.	
CIPHER	Cipher Text Translate2

Availability notes:

- AES CIPHER class keys require z114, z196, or later systems with a CEX3C or later coprocessor with the September 2011 or later licensed internal code.
- AES CIPHER keys can be restricted to be used for cipher text translation only. This support requires zEC12, zBC12, and later systems with a CEX3C, CEX4C, or later coprocessor with September 2012 or later licensed internal code.

MAC keys: Message authentication is the process of verifying the integrity of transmitted messages. Message authentication code (MAC) processing enables you to verify that a message has not been altered. You can use a MAC to check that a message you receive is the same one the message originator sent. The message itself may be in clear or encrypted form.

MAC keys can be used to generate and verify MACs, or can be restricted to just verify MACs.

DES supports the ANSI X9.9-1 procedure, ANSI X9.19 optional double key MAC procedure, and EMV Specification and ISO 16609 for encrypted keys.

DES MAC keys can be used to generate CVVs and CSCs for PIN transactions.

AES supports ciphered message authentication code (CMAC) for encrypted keys and CBC-MAC and XCBC-MAC for clear keys.

HMAC supports FIPS-198 hashed message authentication code (HMAC) for encrypted keys.

Table 6. DES MAC keys

DES keys	Callable services
MAC class (data operation keys): • These keys are used to generate and verify MACs, CVVs, and CSCs. • The keys can be single-length or double-length keys.	
MAC	CVV Key Combine, MAC Generate, MAC Verify, Transaction Validation, VISA CVV Generate, VISA CVV Verify
MACVER	CVV Key Combine, MAC Verify, Transaction Validation, VISA CVV Verify

Table 7. AES MAC keys

AES keys	Callable services
MAC class (data operation keys): • These keys are used to generate and verify MACs. • The keys can be 128, 192, or 256 bits in length. • The key usage flags in the associated data can be used to restrict usage to only generate MACs or to only verify MACs.	
MAC	DK Deterministic PIN Generate, DK PIN Change, DK PAN Modify in Transaction, DK PAN Translate, DK PRW Card Number Update, DK PRW CMAC Generate, DK Random PIN Generate, DK Regenerate PRW, MAC Generate2, MAC Verify2

Availability notes: AES MAC class keys require z114 or z196 systems with a CEX3C coprocessor with the March 2014 or later licensed internal code or zEC12, zBC12, and later systems with a CEX3C, CEX4C, or later coprocessor with March 2014 or later licensed internal code.

Table 8. HMAC MAC keys

HMAC keys	Callable services
MAC class (data operation keys): • These keys are used to generate and verify a keyed hash message authentication code (HMAC). • The keys are variable-length keys (80-2024 bits) and are encrypted under the AES master key. • The key usage flags in the associated data can be used to restrict usage to verify only.	
MAC	HMAC Generate, HMAC Verify, MAC Generate2, MAC Verify2

Availability notes: HMAC keys require z114, z196, or later systems with a CEX3C or later coprocessor with the November 2010 or later licensed internal code.

PIN keys: Personal authentication is the process of validating personal identities in a financial transaction system. The personal identification number (PIN) is the basis for verifying the identity of a customer across the financial industry networks. A PIN is a number that the bank customer enters into an automatic teller machine (ATM) to identify and validate a request for an ATM service.

You can use ICSF to generate PINs and PIN offsets. A PIN offset is a value that is the difference between two PINs. For example, a PIN offset may be the difference between a PIN that is chosen by the customer and one that is assigned by an institution. You can use ICSF to verify the PIN that was generated by ICSF. You can also use ICSF to protect PIN blocks that are sent between systems and to translate PIN blocks from one format to another. A PIN block contains a PIN and non-PIN data. You use PIN keys to generate and verify PINs and PIN offsets, and to protect and translate PIN blocks.

Table 9. DES PIN keys

DES keys	Callable services
PIN class: • These keys are used generate and verify PINs and PIN offsets. • The keys are double-length keys.	
PINGEN	Clear PIN Generate, Clear PIN Generate Alternate, Encrypted PIN Generate, Recover PIN from Offset
PINVER	Encrypted PIN Verify
These keys are used wrap and unwrap PIN blocks:	
IPINENC	Authentication Parameter Generate, Clear PIN Generate Alternate, Encrypted PIN Translate, Encrypted PIN Verify, PIN Change/Unblock, Secure Messaging for PINs
OPINENC	Clear PIN Encrypt, Clear PIN Generate Alternate, Encrypted PIN Generate, Encrypted PIN Translate, PIN Change/Unblock, Recover PIN from Offset

Table 10. AES PIN keys

AES keys	Callable services
PIN class: • These keys are used generate PINs. • The keys can be 128, 192, or 256 bits in length.	
PINCALC	DK Deterministic PIN Generate
These keys are used wrap and unwrap PIN blocks.	
PINPROT	DK Deterministic PIN Generate, DK PAN Translate, DK PIN Change, DK PRW Card Number Update, DK Random PIN Generate, DK Regenerate PRW
These keys are used generate and verify PIN reference words (PRW).	
PINPRW	DK Deterministic PIN Generate, DK PAN Modify in Transaction, DK PAN Translate, DK PIN Change, DK PIN Verify, DK PRW Card Number Update, DK Random PIN Generate, DK Regenerate PRW

Availability notes: AES PIN class keys require z114 or z196 systems with CEX3C with the November 2013 or later licensed internal code or zEC12, zBC12, and later systems with a CEX3C, CEX4C, or later coprocessor with September 2013 or later licensed internal code.

Key-encrypting keys: Key-encrypting keys protect a key that is sent to another system, received from another system, or stored with data in a file. A variation of transport keys are also used to rewrap a key from one key-encrypting key to another key-encrypting key.

Key-encrypting keys are always generated in pairs. Both keys have the same clear key value, but have a different encrypted key value due to the control vector or the associated data.

Exporter key-encrypting key

An exporter key-encrypting key protects keys that are sent from your system to another system. The exporter key at the originator has the same clear value as the importer key at the receiver. An exporter key is paired with an importer key-encrypting key.

DES OKEYXLAT keys must be used when rewrapping a key under a transport key. The AES EXPORTER must have the TRANSLAT key usage enabled when rewrapping a key.

Importer key-encrypting key

An importer key-encrypting key protects keys that are sent from another system to your system. It also protects keys that you store externally in a file that you can import to your system later. The importer key at the receiver has the same clear value as the exporter key at the originator. An importer key is paired with an exporter key-encrypting key.

DES IKEYXLAT keys must be used when rewrapping a key under a transport key. The AES IMPORTER must have the TRANSLAT key usage enabled when rewrapping a key.

Table 11. Keys for the DES key-encrypting key

DES keys	Callable services
Key-encrypting key class: • These keys are used to wrap other keys. • The keys are double-length keys.	
EXPORTER	Control Vector Translate, Data Key Export, ECC Diffie-Hellman, Key Export, Key Generate, Key Test2, Key Test Extended, Key Translate, Key Translate2, PKA Key Generate, PKA Key Translate, Prohibit Export Extended, Remote Key Export, Secure Messaging for Keys, Symmetric Key Generate, TR-31 Export, TR-31 Import, Unique Key Derive
IMPORTER	Control Vector Translate, Data Key Import, ECC Diffie-Hellman, Key Generate, Key Import, Key Test2, Key Test Extended, Key Translate, Key Translate2, Multiple Secure Key Import, PKA Key Generate, PKA Key Import, PKA Key Translate, Prohibit Export Extended, Remote Key Export, Restrict Key Attribute, Secure Key Import, Secure Messaging for Keys, Symmetric Key Generate, TR-31 Export, TR-31 Import
IMP-PKA	PKA Key Import, Remote Key Export, Trusted Block Create
IKEYXLAT, OKEYXLAT	Control Vector Translate, Key Translate, Key Translate2, TR-31 Export,TR-31 Import

Table 12. Keys for the AES key-encrypting key

AES keys	Callable services
Key-encrypting key class: • These keys are used to wrap other keys. • The keys can be 128, 192, or 256 bits in length.	
EXPORTER	ECC Diffie-Hellman, Key Generate2, Key Test2, Key Translate2, PKA Key Generate, PKA Key Translate, Symmetric Key Export
IMPORTER	ECC Diffie-Hellman, Key Generate2, Key Test2, Key Translate2, PKA Key Generate, PKA Key Import, PKA Key Translate, Restrict Key Attribute, Secure Key Import2, Symmetric Key Import2

Availability notes: AES key-encrypting class keys require z114, z196, or later systems with a CEX3C or later coprocessor with the September 2011 or later licensed internal code.

Key-generating keys: Key-generating keys are used to derive unique-key-per transaction keys.

Table 13. DES key-generating keys

DES keys	Callable services
Key-generate key class: • These keys are used to derive keys. • The keys are double-length keys. • The key usage flags in the control vector determine which services the KEYGENKY key may be used with.	
KEYGENKY	Diversified Key Generate, Encrypted PIN Translate, Encrypted PIN Verify, Unique Key Derive
DKYGENKY	Diversified Key Generate, PIN Change/Unblock

Table 14. AES key-generating keys

AES keys	Callable services
Key-generate key class: • These keys are used to derive keys. • The keys can be 128, 192, or 256 bits in length.	
DKYGENKY	Diversified Key Generate2

Availability notes: AES key-generating class keys require z114 or z196 systems with a CEX3C coprocessor with the November 2013 or later licensed internal code, or zEC12, zBC12, and later systems with a CEX3C, CEX4C, or later coprocessor with September 2013 or later licensed internal code.

Cryptographic variable keys: These DES keys are used to encrypt special control values in DES key management.

Table 15. DES cryptographic variable keys

DES keys	Callable services
Cryptographic-variable class: • These keys are used in the special verbs that operate with cryptographic variables. • The keys are single-length keys.	
CVARENC	Cryptographic Variable Encipher
CVARXCVL	Control Vector Translate
CVARXCVR	Control Vector Translate

Secure messaging keys: These keys are used to encrypt keys and PINs for incorporation into a text block. The text block is then encrypted to preserve the security of the key value. The encrypted text block, normally the value field in a TLV item, can be incorporated into a message sent to an EMV smart card.

Table 16. DES secure messaging keys

DES keys	Callable services
Secure-messaging class (data operation keys) These keys are used to encrypt keys or PINs. The keys are double-length keys. The key usage flags in the control vector determine which services the key may be used with.	
SECMSG	Diversified Key Generate, Secure Messaging for Keys, Secure Messaging for PINs

Table 17. AES secure messaging keys

AES keys	Callable services
Secure-messaging class (data operation keys) These keys are used to encrypt keys or PINs. The keys can be 128, 192, or 256 bits in length.	
SECMSG	DK PIN Change

Asymmetric keys

ICSF supports RSA and ECC keys:

RSA

An RSA key pair includes a private key and a public key. RSA keys can be used for key distribution and authentication. The private key can be restricted to authentication only or key management only.

Table 18. RSA keys

Key	Callable services
The length of the modulus may be 512-4096 bits. Modules-exponent and Chinese Remainder Theorem formats are supported.	
Private	Digital Signature Generate, Key Test2, PKA Public Key Extract, Public Key Decrypt, Restrict Key Attribute, SET Block Decompose, Symmetric Key Import, Symmetric Key Import2
Public	Digital Signature Verify, Key Test2, Public Key Encrypt, SET Block Compose, Symmetric Key Export, Symmetric Key Export with Data, Symmetric Key Generate

Availability notes: RSA keys with a modulus greater than 2048 bits are supported on the z9 EC, z9 BC, and later systems with a CEX2C or later coprocessor with the November 2007 or later licensed internal code.

ECC

An ECC key pair includes a private and public key. ECC keys can be used for authentication and symmetric key derivation. ECC keys are used to derive AES and DES keys using the Diffie-Hellman protocol. The private key can be restricted to authentication only or key derivation only.

Table 19. ECC keys

Key	Callable services
Private	Digital Signature Generate, ECC Diffie-Hellman
Public	Digital Signature Verify, ECC Diffie-Hellman

Availability notes: ECC keys are supported on the z10 EC, z10 BC, and later systems with a CEX3C and later coprocessor with the November 2010 or later licensed internal code.

Trusted blocks

Trusted blocks are used in remote key management for ATMs and other remote devices.

Table 20. Trusted blocks

Key	Callable services
Inactive	Trusted Block Create
Active	Remote Key Export,Trusted Block Create, PKA Key Import

Availability notes: Trusted blocks are supported on the z9 EC, z9 BC, and later servers with a CEX2C and later coprocessor with the May 2006 or later licensed internal code.

Regional cryptographic operational keys

These keys are used only with the ICSF PKCS #11 services and only when a regional cryptographic server is online and active. Regional cryptographic servers are network-attached, standalone devices or dedicated Linux LPARs that perform geography-specific cryptography. Currently, the geography-specific cryptography algorithms supported by these devices are the Chinese symmetric algorithm SM4, the asymmetric algorithm SM2, and the hashing algorithm SM3. The supported key types are SM2 and SM4.

For more information about the PKCS #11 services' support for regional cryptographic server keys, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* and *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

PKCS #11 operational keys

These keys are used only with the ICSF PKCS #11 services. The following types are supported:

- Symmetric key types: DES, TDES, AES, BLOWFISH, and RC4
- Asymmetric key types: RSA, DSA, ECDSA, DH, and ECDH

- HMAC key types: Generic Secret

For more information about PKCS #11 and keys, seez/*OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

Protection and control of cryptographic keys

Because the cryptographic algorithms are all key-controlled algorithms, the security of protected data depends on the security of the cryptographic key. With the exception of master keys, which are physically secured, keys that require a high level of protection are enciphered under another key to provide this necessary security.

A key can be protected under either a master key, a transport key, or a PKA key. The master key protects a key you use on the system. When you send a key to another system, you protect it under a transport key rather than under the master key. You can use RSA public keys to protect DES, AES, and HMAC keys that are transported between systems.

ICSF controls the use of AES and DES keys by separating them into types that can be used to do only specific functions.

Master key concept

ICSF uses the master key concept to protect cryptographic keys. Master keys, which are stored in secure hardware in the cryptographic feature, are used to encrypt all other keys on the system. All other keys that are encrypted under these master keys are stored outside the protected area of the cryptographic feature. This is an effective way to protect a large number of keys while needing to provide physical security for only a few master keys.

The master keys are used only to encipher and decipher keys. Other key-encrypting keys that are called *transport keys* also encipher and decipher keys and are used to protect cryptographic keys you transmit to other systems. These transport keys, while on the system, are also encrypted under a master key.

Symmetric key separation

The cryptographic features controls the use of AES and DES keys by separating them into unique types. How a key is used distinguishes it from other keys. The cryptographic feature allows you to use only a specific type of key for its intended purpose. For example, a key that is used to protect data cannot be used to protect a key.

DES keys

Key separation for DES keys is controlled by the control vector. The control vector has fields for the key type, key usage, and key management. See Appendix B. Control Vectors and Changing Control Vectors with the CVT Callable Service in *z/OS Cryptographic Services ICSF Application Programmer's Guide* for details on control vectors. The control vector is cryptographically bound to the encrypted key value when the key is encrypted under the master key or a transport key.

The cryptographic coprocessor encrypts each operational key under a unique variation of the DES master key. Each variation encrypts a different type of key. Although you define only one master key, in effect you have a unique master key to encrypt each type of key that is used in DES services.

DES keys can be single-length or double-length keys, depending on their key type. A single-length key is 64 bits and a double-length key is 128 bits. For double-length keys, one control vector exists for the left half of the key and another control vector for the right half of the key. Therefore, ICSF creates a master key variant or transport key variant for each half of the key the master key or transport key will protect. Triple-length DATA keys do not have a control vector.

A key that is protected under the master key is in *operational form*, which means that ICSF can use it in cryptographic functions on the system. As is shown in Figure 1, all secure keys that you want ICSF to use in cryptographic functions are enciphered under the master key.

Whenever the master key is used to encipher a key, the cryptographic coprocessor produces a variation of the master key according to the type of key that is being enciphered. These variations are called *master key variants*. The cryptographic coprocessor creates a master key variant by exclusive ORing a fixed pattern, called a *control vector*, with the master key. Each type of key that is used in DES services has a unique control vector associated with it. For example, the cryptographic coprocessor uses one control vector when the master key enciphers a PIN generation key and a different control vector when the master key enciphers a PIN verification key.

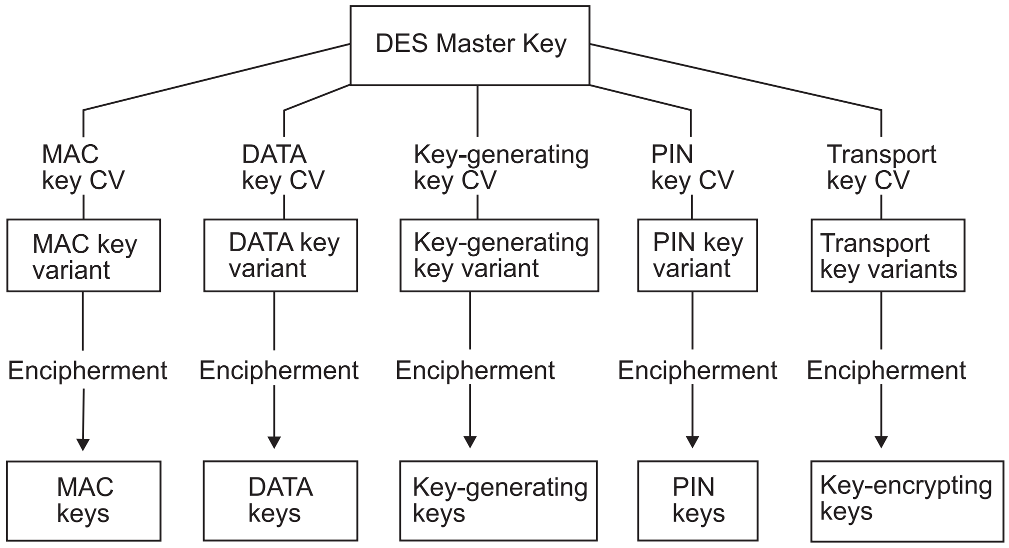


Figure 1. Keys protected in a system

When systems want to share keys, transport keys can be used to protect keys sent outside of systems. A key that is enciphered under a transport key cannot be used in a cryptographic function. The key must first be brought into a system, deciphered from under the transport key, and enciphered under the system's master key.

ICSF creates variations of a transport key to encrypt a key according to its type. Whenever a transport key is used to encipher a key, the cryptographic feature produces the variation of the transport key according to the type of key that is being enciphered. This allows for key separation when a key is transported off the system.

A transport key variant, also called a *key-encrypting key variant*, is created in the same way as a master key variant. The transport key is exclusive ORed with a control vector that is associated with the key type of the key it protects. See Appendix B, “Control Vector Table,” on page 395 for a listing of the control vector that is used for each key type.

AES and HMAC keys

AES and HMAC key separation is controlled by the associated data section in the key token. The associated data section contains fields for type of algorithm for which the key can be used, key type, key usage, and key management. In addition to the algorithm and key type, the values of the key-usage and key-management fields further restrict the use of a key.

The associated data is cryptographically bound to the key token when the key value is encrypted under the master key or a transport key.

Asymmetric key usage

RSA keys can be restricted for symmetric key distribution and authentication usage. ECC keys can be restricted for authentication and symmetric key derivation usage.

Migrating from PCF and CUSP key types

Your installation may use Programmed Cryptographic Facility (PCF) or Cryptographic Unit Support Program (CUSP). ICSF provides key types that are similar to the PCF and CUSP key types and provides other key types for enhanced key separation and more functions. You cannot use a PCF or CUSP key on ICSF, but you can convert a PCF or CUSP key into an ICSF key. Table 21 lists which ICSF key types correspond to the PCF and CUSP key types.

Table 21. PCF and CUSP and their corresponding ICSF key types

PCF and CUSP key type	ICSF key type
Local key	Exporter key-encrypting key or Output PIN-encrypting key
Remote key	Importer key-encrypting key or Input PIN-encrypting key
Cross key	Importer key-encrypting key and exporter key-encrypting key or Input PIN-encrypting key and output PIN-encrypting key

ICSF provides compatibility modes and a conversion program to help you run PCF or CUSP with ICSF and to migrate from PCF or CUSP to ICSF. The conversion program converts PCF and CUSP keys to ICSF keys. For information about migration from PCF or CUSP to z/OS ICSF, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Key strength and wrapping of key

Key strength can be measured as “bits of security” as described in the documentation of NIST and other organizations. Each individual key will have its “bits of security” computed, then the different key types (AES, DES, ECC, RSA, HMAC) can then have their relative strengths compared on a single scale. When the raw value of a particular key falls between discreet values of the NIST table the lower value from the table will be used as the “bits of security”.

The following tables show some examples of the restrictions due to key strength. When wrapping an HMAC key with an AES key-encrypting key, the strength of the AES key-encrypting key depends on the attributes of the HMAC key.

Table 22. AES EXPORTER strength required for exporting an HMAC key under an AES EXPORTER

Key-usage field 2 in the HMAC key	**Minimum strength of AES EXPORTER to adequately protect the HMAC key**
SHA-256, SHA-384, SHA-512	256 bits
SHA-224	192 bits
SHA-1	128 bits

Table 23. Minimum RSA modulus length to adequately protect an AES key

Bit length of AES key to be exported	**Minimum strength of RSA wrapping key to adequately protect the AES key**
128	3072
192	7860
256	15360

Access control points

In order to comply with cryptographic standards, including ANSI X9.24 Part 1 and PCI-HSM, ICSF will provide a way to ensure that a key is not wrapped with a key weaker than itself. ICSF will provide a set of access control points in the ICSF role to control the wrapping of keys. ICSF administrators can use these access control points to meet the customers individual requirements.

There are new and existing access control points that control the wrapping of keys by master and key-encrypting keys. These ACP will either prohibit the wrapping of a key by a key of weaker strength or warning (return code 0, reason code non-zero) when a key is wrapped by a weaker key. All of these access control points are disabled by default in the ICSF role.

The processing of callable services will be affected by these access control points. Here is a description of the access control points, the wrapping it controls and the affect on services. These access control points apply to symmetric and asymmetric keys.

When the **Prohibit weak wrapping - Transport keys** access control point is enabled, any service that attempts to wrap a key with a weaker transport key will fail.

When the **Prohibit weak wrapping - Master keys** access control point is enabled, any service that wraps a key under a master key will fail if the master key is weaker than the key being wrapped.

When the **Warn when weak wrap - Transport keys** access control point is enabled, any service that attempts to wrap a key with a weaker transport key will succeed with a warning reason code.

When the **Warn when weak wrap - Master keys** access control point is enabled, any service that attempts to wrap a key with a weaker master key will succeed with a warning reason code.

24-byte DATA keys with a zero control vector can be wrapped with a 16-byte key, the DES master key or a key-encrypting key, which violates the wrapping requirements. The **Prohibit weak wrapping – Transport keys** and **Prohibit weak wrapping – Master keys** ACPs do not cause services to fail for this case. The **Disallow 24-byte DATA wrapped with 16-byte Key** ACP does control this wrapping. When enabled, services will fail. The **Warn when weak wrap – Transport keys** and **Warn when weak wrap – Master keys** ACPs will cause the warning to be returned when the ACPs are enabled.

When the **TBC – Disallow triple-length MAC key** ACP is enabled, CSNDRKX will fail to import a triple-length MAC key under a double-length key-encrypting key. CSNBTBC will not wrap a triple-length MAC key under a double-length key-encrypting key. The **Prohibit weak wrapping – Transport keys** and **Prohibit weak wrapping – Master keys** ACPs do not cause services to fail for this case. The **Warn when weak wrap – Transport keys** and **Warn when weak wrap – Master keys** ACPs will cause the warning to be returned when the ACPs are enabled.

If the **Prohibit Weak Wrap** ACP is enabled, RSA private keys may not be wrapped using a weaker DES key-encrypting key. Enabling the **Allow weak DES wrap of RSA private key** ACP will override this restriction.

DES key wrapping

ICSF wraps the key value in a DES key token using one of two possible methods:

- The original method of DES key wrapping has been used by ICSF since its initial release. Using this original key wrapping method, the key value in DES tokens are encrypted using triple DES encryption, and key parts are encrypted separately.
- The enhanced method of symmetric key wrapping, introduced in ICSF FMID HCR7780, is designed to be ANSI X9.24 compliant. Using the enhanced method, the key value for keys is bundled with other token data and encrypted using triple DES encryption and cipher block chaining mode. The enhanced method is available on z196, z144, and later servers with a CEX3 or later coprocessor with the November 2010 or later licensed internal code.

Using the DEFAULTWRAP keyword in the installation options data set, you can specify the default wrapping method that ICSF will use for internal key tokens and external key tokens. The default wrapping method for internal key tokens and the default wrapping method for external key tokens are independent to each other and are specified separately. If the installation options data set does not contain the DEFAULTWRAP keyword, the original method of symmetric key wrapping will be the default key wrapping method for both internal and external key tokens. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for information on the installation options data set and the DEFAULTWRAP keyword.

If you are sharing a CKDS with a release of ICSF that does not support the enhanced wrapping method (HCR7770 and earlier), you should use the original wrapping method until all systems sharing the CKDS support the enhanced wrapping method.

A CKDS conversion utility, CSFCNV2, enables you to convert all tokens in the CKDS to use either the original or the enhanced wrapping method. See Chapter 20, "Rewrapping DES key token values in the CKDS using the utility program CSFCNV2," on page 365 for more information.

AES key wrapping

The AESKW wrapping method for AES keys in the variable-length symmetric key tokens is defined in standard ANSI X9.102.

DES master key

Since ICSF only allows a 16-byte DES master key to be loaded, ICSF cannot be compliant for key strength for 24-byte operational keys wrapped by the DES master key. Starting with ICSF FMID HCR77A0, a 24-byte master key can be loaded. Only cryptographic coprocessors with the October, 2012 licensed internal code support this key length. The **DES master key – 24-byte key** access control point must be enabled in the ICSF role. See Chapter 8, "Managing PKCS #11 master keys," on page 159 for more details.

Protection of distributed keys

When you store a key with a file or send it to another system, you can protect the key in either of these ways:

- DES keys in a CCA key token enciphered under a DES transport key.
- DES keys in a TR-31 key block enciphered under a DES transport key.
- AES and HMAC keys in a CCA key token enciphered under an AES transport key.
- AES, DES, and HMAC keys enciphered under the a RSA public key.
- RSA private keys in a CCA key token enciphered under a DES or AES transport key.
- RSA private keys in a smart card format enciphered under a DES transport key.
- ECC private keys in a CCA key token enciphered under an AES transport key.

When ICSF enciphers a key under a transport key, the key is not in operational form and cannot be used to perform cryptographic functions. When you receive a key from a system, the key is enciphered under a transport key. You can reencipher the key from under the transport key to under your master key. You can then use the key on your system. When a key is enciphered under a transport key, the sending system considers it in exportable form, and the receiving system considers it in importable form. When a key is reenciphered from under a transport key to under a system's master key, it is in operational form again.

In an RSA public key cryptographic system, the sending system and receiving system do not need to share complementary importer and exporter key pairs to exchange data-encrypting keys. The sender uses the receiver's public key to encipher the data-encrypting key. The receiver uses his or her own private key to decipher the data-encrypting key. You can use RACF to control which applications can use specific keys and services. For more information, see "System authorization facility (SAF) controls" on page 81.

Protecting keys stored with a file

You may want to store encrypted data in a file that is stored on DASD or on magnetic tape. For example, if you use a data-encrypting key to encrypt data in a file, you can store the data-encrypting key with the encrypted data. As is shown in Figure 2 on page 27, you use an importer key-encrypting key to encrypt the data-encrypting key.

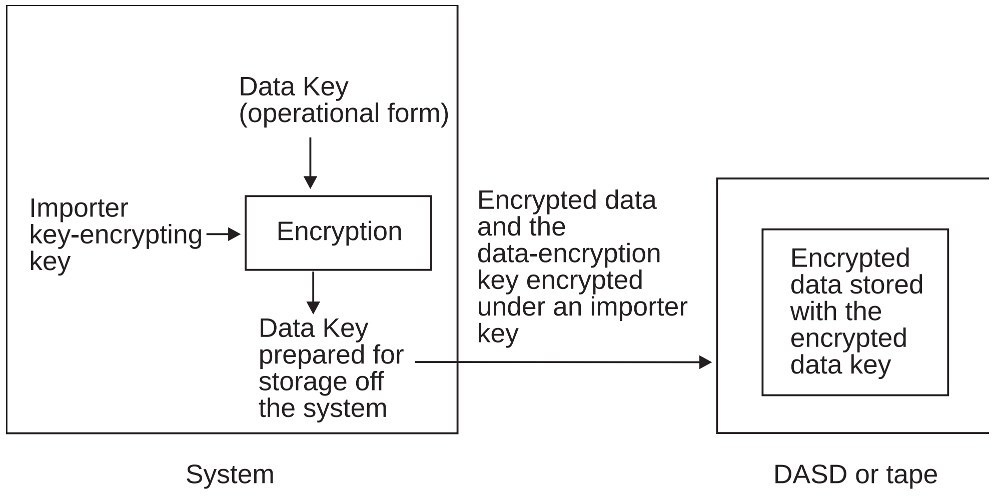

Data Key
(operational form)
Importer key-encrypting key
Encryption
Data Key prepared for storage off the system
Encrypted data and the data-encryption key encrypted under an importer key
Encrypted data stored with the encrypted data key
System
DASD or tape

Figure 2. Keys protected in a file outside the system

When you encipher a key under an importer key, the key is no longer enciphered under the master key and is no longer operational. You can store the key off the system because the key will not become obsolete if you change the master key. The importer key that protects the data-encrypting key is reenciphered under the correct master key during a master key change. Therefore, when enciphered under the importer key, the data-encrypting key is not directly affected by a master key change.

When you are ready to use the data-encrypting key, use ICSF to reencipher it from under the transport key to under the master key. This makes the data-encrypting key operational. You can then use the data-encrypting key to decrypt the data.

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 an earlier 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 - Create Block in inactive form.
- Trusted Block Create - Activate an inactive block.
- PKA Key Import - Import an external trusted block.

- Remote Key Export - Gen or export a non-CCA node key.

Using DES and AES transport keys to protect keys sent between systems

You can send and receive keys and PINs between your system and another system. For example, if you send encrypted data to another system, you also send the data-encrypting key that enciphered the data. The other system can then use the data-encrypting key to decipher the data. In a financial system, you might need to send a PIN from the system that received the PIN from a customer to a system that uses it to verify a customer's identity. As shown in Figure 3, when you send the PIN between systems, you encipher the PIN under a PIN-encrypting key.

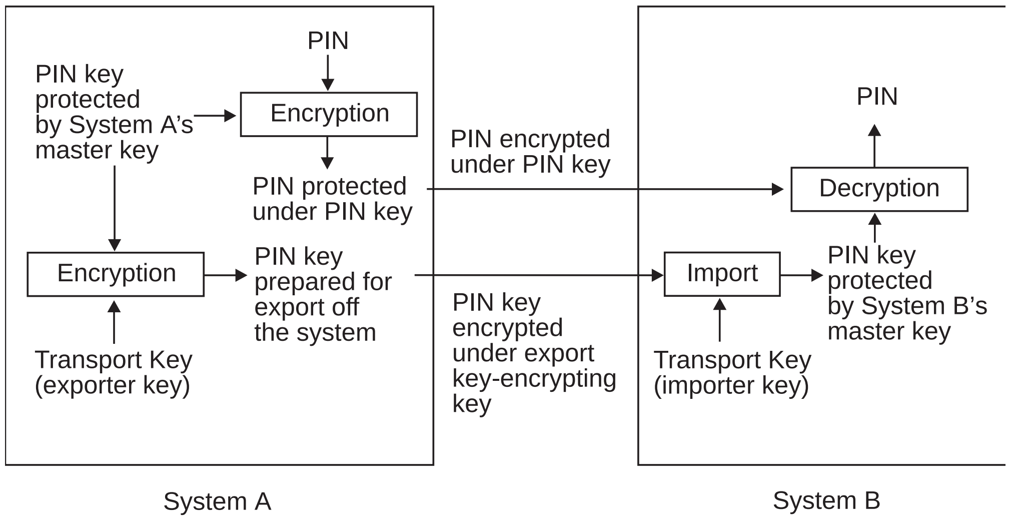

PIN
PIN key protected by System A's master key
Encryption
PIN protected under PIN key
Encryption
PIN key prepared for export off the system
Transport Key (exporter key)
PIN encrypted under PIN key
PIN key encrypted under export key-encrypting key
PIN
Decryption
Import
PIN key protected by System B's master key
Transport Key (importer key)
System A
System B

Figure 3. Keys and PINs protected when sent between two systems

Two systems do not share a master key. When you send a key to another system, you do not encrypt it under a master key. You encrypt it under a transport key.

Two systems that exchange keys share transport keys that have the same clear value. At the sending system, the transport key is an exporter key-encrypting key. At the receiving system, the transport key is an importer key-encrypting key. When the sending system wants to send a key, the sending system encrypts the key under an exporter key-encrypting key. The key is in exportable form on the system that sends the key.

The key is in importable form on the system that receives the key. The receiving system reencrypts the key from under the importer key-encrypting key to under its own master key. The key is then in operational form and can be used on the system.

Using RSA public keys to protect keys sent between systems

The ability to create more-secure key-exchange systems is one of the advantages of combining DES or AES and PKA support in the same cryptographic system. Because PKA cryptography is more computationally intensive than symmetric cryptography, it is not the method of choice for all cryptographic functions. It can be used, however, in combination with symmetric cryptography to enhance the security of key exchange. Symmetric keys can be exchanged safely between two systems when encrypted using an RSA public key. Sending system and receiving system do not need to share a secret key to be able to exchange RSA-encrypted symmetric keys. An example of this is shown in Figure 4. The sending system enciphers the symmetric key under the receiver's RSA public key and sends the enciphered symmetric key to the receiver. The receiver uses his or her RSA private key to decipher the symmetric key.

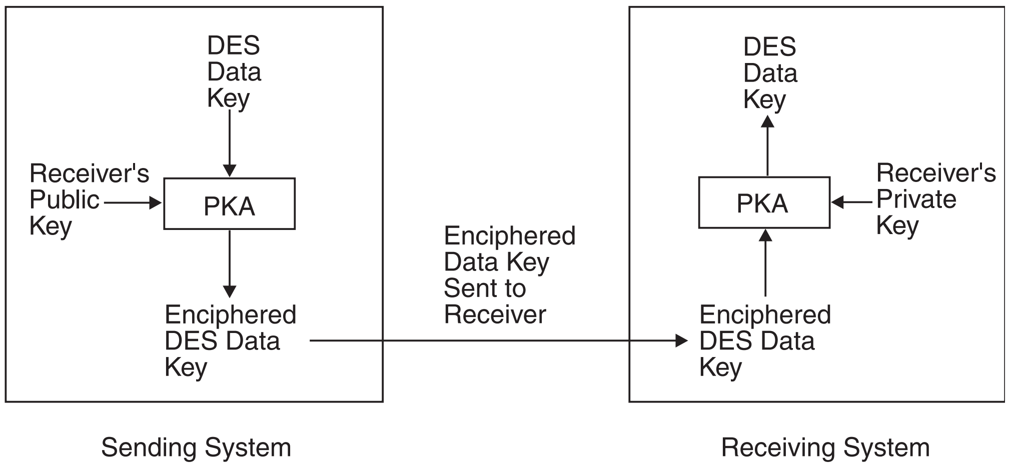


Figure 4. Distributing a DES data-encrypting key using an RSA cryptographic scheme

Not all symmetric keys can be wrapped using an RSA key. AES and DES data-encryption keys are supported (fixed-length format key token) as well as AES and HMAC keys (variable-length format key token).

Protection of data

You use data-encrypting keys to encrypt data. On a system, a data-encrypting key is often encrypted under the master key.

A data-encrypting key can encrypt data that is stored in a file outside the system. The data-encrypting key itself is encrypted under a transport key.

You may also need to protect data that you send from one system to another system. The data-encrypting key that protects this data must be sent with the data so that the receiving system can decrypt the data. In this case, the data-encrypting key is encrypted under a transport key.

Sometimes two systems that want to exchange data are not directly connected. There may be intermediate systems between the systems that the data must travel through, as in Figure 5 on page 30.

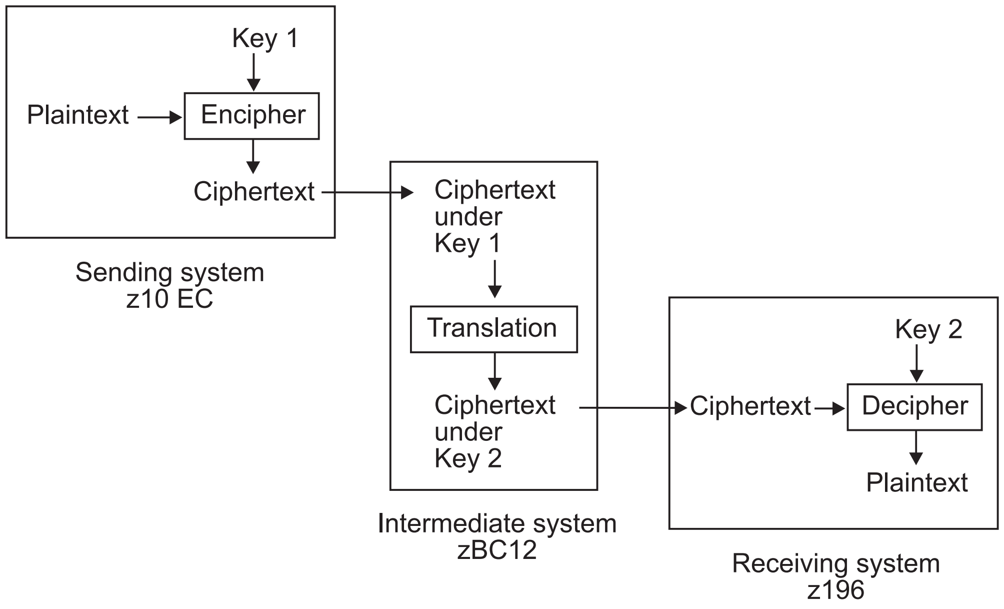


Figure 5. Data protected when sent between intermediate systems

In this situation, when you pass enciphered data to a system, you do not send a data-encrypting key to decipher the data at the receiving system. Instead, the systems establish pairs of data-encrypting and cipher text-translation keys that exist on the systems. These keys encipher and reencipher the data. The data ends up enciphered under a data-encrypting key that exists on the receiving system. Transport keys may be needed to establish the data-encrypting keys and the cipher text-translation keys on the systems.

Both the sending and receiving systems give data-translation keys to the intermediate system. On the intermediate system, a data-translation key from the sending system matches a data-encrypting key on the sending system. In Figure 5, this key is called *Key 1*. Also on the intermediate system, a data-translation key from the receiving system matches the data-encrypting key on the receiving system. In Figure 5, this key is called *Key 2*. Note that *Key 1* and *Key 2* do not have the same clear key value.

The cipher text-translation keys cannot decipher data. They are used in the ciphertext translate callable service, which reenciphers data from protection under one key to protection under another key.

On the sending system, the plaintext is enciphered under *Key 1*, so it is ciphertext. Then the ciphertext is sent to the intermediate system. At the intermediate system, the data is reenciphered from under *Key 1* to under *Key 2* without appearing as plaintext. When the receiving system receives the ciphertext, the system can decipher the ciphertext from under *Key 2*, so it is plaintext.

Cipher text-translation keys are also used when there is more than one intermediate system between the sending system and receiving system. The sending system and the first intermediate system share a data-encrypting/cipher text-translation key pair. Each pair of neighboring intermediate systems shares a data-translation key pair. The final intermediate system and the receiving system share a cipher text-translation/data-encrypting key pair.

Chapter 3. Managing cryptographic keys

To perform cryptographic services, you need to know how to create, maintain, and use cryptographic keys. This topic discusses CCA key in detail and gives an overview for PKCS #11 keys. For a detail description of PKCS #11, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

There are three types of cryptographic key data sets:

- Cryptographic key data set (CKDS).
- Public key data set (PKDS).
- PKCS #11 token data set (TKDS).

See Chapter 4, “Setting up and maintaining cryptographic key data sets,” on page 69 for additional details.

Managing CCA cryptographic keys

To perform cryptographic services, you need to know how to create, maintain, and use cryptographic CCA keys. This topic gives an overview on entering master keys, generating keys, entering keys into the cryptographic key data set (CKDS) and the public key data set (PKDS), and distributing keys.

Generating cryptographic keys

Symmetric keys

Using ICSF, you can generate DES and AES keys by using either the key generator utility program (KGUP) or the key generate callable service. KGUP stores the key that it generates in the CKDS. The key generate callable service returns the key to the application program that called it instead of storing it in the CKDS. The application program can then call the CKDS key record write service to store the key in the CKDS.

When you use callable services to generate keys, you pass parameters that specify information about the key you want generated. The services generates keys in these possible forms:

- Operational, if the master key protects it.
- Importable, if an importer key-encrypting key protects it.
- Exportable, if an exporter key-encrypting key protects it.

Key Generator Utility Program (KGUP): You can use KGUP to generate DES and AES keys in either an operational form or an exportable form. When KGUP generates a key in the operational form, it stores it in the CKDS. When KGUP generates a key in exportable form, you can send it to another system.

To specify the function that you want KGUP to perform, you use KGUP control statements. For a detailed description of how to use the program to generate keys, see Chapter 11, “Managing Cryptographic Keys Using the Key Generator Utility Program,” on page 185.

Key generate callable service: The key generate callable service generates a single DES or AES key or a pair of DES keys. The Key Generate2 callable service generates a single AES or HMAC key or a pair of AES or HMAC keys. Unlike

KGUP, the key generate callable service does not store the keys in the CKDS, but returns them to the application program that called the service. The application program can then call the dynamic CKDS update service to store the keys in the CKDS.

Use of these callable services is optional and should be enabled as required for authorized usage. Enabling these callable services is not recommended for production and usage requires special consideration.

For more information about these callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Services that import clear key values: There are several services that accept a clear key value and return an operational key. These services can be used to generate keys within an application:
- Clear Key Import
- Multiple Clear Key Import
- Key Part Import
- Key Part Import2
- Multiple Secure Key Import
- Secure Key Import
- Secure Key Import2

The clear key import services takes a single clear key value and returns an operational DES DATA key. The multiple clear key import services take a single clear key value and return an operational AES or DES DATA key.

The key part import services allow applications to create a key using key parts. The key parts are combined together to form a complete key. The key part import service is used with DES keys and the Key Part Import2 service is used with AES keys. A key in the CKDS can be used by these services. The key part bit must be enable in the DES control vector or the AES key associated data for the key to be processed by these services.

The secure key import services are used to create a key from a single clear key part. The key can be in operational or importable form. The multiple secure key import and secure key import services support DES and AES keys in the fixed-length format token. The Secure Key Import2 service supports AES keys in the variable-length format token. Note that the Special Secure Mode control must be enabled to use these services.

The use of these callable services is optional and should be enabled as required for authorized usage. Enabling these callable services is not recommended for production and usage requires special consideration.

For more information about these callable service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Enhanced key management for crypto assist instructions: To exploit clear key DES and AES instructions on the CPACF, ICSF can generate and format clear DES and AES tokens with a clear key value to be used in callable services and stored in the cryptographic key data set (CKDS). With clear key support on the CKDS, clear keys do not have to appear in application storage during use. Clear key tokens on the CKDS can be referenced by label name in these callable services:

- Symmetric Key Encipher
- Symmetric Key Decipher
- Symmetric MAC Generate
- Symmetric MAC Verify

On systems sharing the CKDS without this support, it is highly recommended that you SAF-protect the label name of the clear key tokens on the other systems. This will provide additional security for your installation. See "System authorization facility (SAF) controls" on page 81 for more information.

Encrypted key support for Crypto Assist instructions: ICSF will exploit the performance of the CP Assist for Cryptographic Functions using encrypted AES and DES keys stored in the CKDS. Symmetric Key Encipher and Symmetric Key Decipher callable services will accept the label of a encrypted key as the key identifier. For more information about encryption using protected-key CPACF, see *z/OS Cryptographic Services ICSF Overview*.

Asymmetric keys

RSA and ECC keys can be generated using the PKA Key Generate service.

The RSA private keys can be generated within the secure boundary of the card and never leave the secure boundary. Only the domain that created the retained key can access it. For more information on how to retain a generated key, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Normally, the output key is randomly generated. You may find it useful in testing situations to re-create the same key values. By providing regeneration data, a seed can be supplied so that the same value of the generated key can be obtained in multiple instances. To generate the keys based on the value supplied in the regeneration_data parameter, you must enable one of these access control points:

- When using the RETAIN keyword, enable the **PKA Key Generate - Permit Regeneration Data Retain** access control point.
- When not using the RETAIN keyword, enable the **PKA Key Generate - Permit Regeneration Data** access control point.

For more information on enabling access control points, refer to *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Entering keys

This topic gives you an overview of key entry and the methods of key entry.

Master keys are used to protect sensitive cryptographic keys that are active on your system. The number and types of master keys you need to enter depends on your hardware configuration and application requirements:

- The DES master key (DES-MK) protects DES keys.
- The RSA master key (RSA-MK) protects RSA keys.
- The AES master key (AES-MK) protects AES keys and HMAC keys.
- The ECC master key (ECC-MK) protects ECC and RSA keys.

The first time you start ICSF on your system, you may enter master keys and initialize the CKDS and PKDS. You can then generate and enter the keys you use to perform cryptographic functions. The master keys you enter protect sensitive keys stored in the CKDS and PKDS.

If you have no coprocessor, you can initialize the CKDS for use with clear AES and DES data keys. This CKDS cannot be used on a system with cryptographic coprocessors.

Because master key protection is essential to the security of the other keys, ICSF stores the master keys within the secure hardware of the cryptographic coprocessors. This nonvolatile key storage area is unaffected by system power outages because it is protected by a battery power unit. The values of the master keys never appear in the clear outside the cryptographic coprocessors.

Managing master keys involves these tasks:
- Entering the master keys the first time you start ICSF.
- Reentering the master keys if they are cleared.
- Changing master keys periodically.

Entering master keys

The types of master keys you can enter and the steps you take to enter master keys depend on your system processor and hardware features.

The following methods may be used to enter master keys:
- Pass phrase initialization

  The pass phrase initialization utility allows the user of ICSF to set all of the CCA master keys available on their systems and initialize the CKDS and PKDS. For steps in using the pass phrase initialization utility, see Chapter 6, “Using the pass phrase initialization utility,” on page 95.
- Master Key Entry panels

  The Master Key Entry panels are enhanced ISPF panels enabling you to enter master key parts in the clear. Use these panels to enter master key parts into CCA coprocessors. The master key parts appear briefly in the clear in MVS host storage within the address space of the TSO user before being transferred to the secure hardware. Within the boundaries of the secure hardware, the key parts are combined to produce the master key. The master key part entry panels provide a level of security for master key entry that is superior to that provided with PCF. Master key part entry is provided for installations where the security requirements do not warrant the additional expense of the optional TKE workstation. For master key entry steps on the coprocessors, see Chapter 7, “Managing CCA Master Keys,” on page 103.
- Trusted Key Entry (TKE) workstation

  The TKE workstation is an optional hardware feature. The TKE workstation uses a variety of public key cryptographic techniques to ensure both the integrity and privacy of the logically secure master key transfer channel. You can use a single TKE workstation to set up master keys in all CCA and EP11 coprocessors within a server complex.
  - TKE must be used to enter P11 master keys on a PKCS #11 cryptographic coprocessor.

  For information on using the TKE workstation, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Servers or processor models may have multiple cryptographic coprocessors. The master keys must be the same for all coprocessors accessed by the same operating system.

When you have entered symmetric master keys, go to the Key Data Set Management panel to:

- Create the CKDS header record.
- Activate the DES master key and/or AES master key and read the CKDS into storage.
- Create system keys that ICSF uses for internal processing and read the CKDS into storage again.

When you have entered the asymmetric master keys, go to the Key Data Set Management panel to initialize the PKDS. This process will:

- Create the PKDS header record.
- Activate the RSA master key and/or the ECC master key and read that PKDS into storage.

When you have entered the P11 master keys, select option 1 from the ICSF TKDS Master Key Management panel to initialize the TKDS. This process will:

- Create the TKDS header record.
- Activate the P11 master key.

Entering system keys into the CKDS

The ICSF CKDS has several sets of system keys. These are the keys with labelname of X'00' and are installed during CKDS initialization.

The system keys are required in the fixed-length format CKDS with record authentication enabled. These keys are created when the CKDS is initialized. If record authentication is not enabled, no system keys are created when the CKDS is initialized.

There are no system keys required for the variable-length format CKDS. Other existing system keys in a CKDS initialized on an older server are not used and their presence in the CKDS has no effect on operations.

Entering keys into the CKDS

All DES, AES, and HMAC keys (except for master keys) can be stored in the CKDS.

There are several methods you can use to enter keys into the CKDS:

- Key generator utility program (KGUP)

  You can use KGUP to enter keys into the CKDS.
- Dynamic CKDS update callable services

  You can program applications to use the CKDS key record create service to create new entries in the CKDS and use the CKDS key record write service to enter key tokens into the CKDS.
- Trusted Key Entry (TKE) workstation

  DES operational key support is available for all CCA Cryptographic coprocessors. AES operational key support is available for CCA Cryptographic coprocessors that are a CEX2C and later. You can load key parts for all operational keys into key part registers on the card. To load the accumulated key into the CKDS, you must use the ICSF Operational Key Load panel or KGUP. For more information, refer to the *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.
- Enterprise Key Management Foundation (EKMF)

The Enterprise Key Management Foundation (EKMF) provides online key management to ICSF as well as to IBM cryptographic products on other platforms. EKMF offers centralized key management for CCA symmetric and asymmetric keys and for certificates. EKMF automates the key management process and exchanges and replaces keys and certificates on demand. Also, to assure continuous operation, EKMF maintains backup copies of all critical keys.

For additional information, contact the Crypto Competence Center at **ccc@dk.ibm.com** or at: **https://www-304.ibm.com/jct05001c/dk/security/cccc/**.

The table in Table 24 shows which keys can be entered by each of these methods.

Table 24. Methods for entering each key type into the CKDS

Key Type	KGUP	Dynamic Update	TKE	EKMF
Data-encrypting (DES DATA)	X	X		X
Data-encrypting (AES and DES CIPHER)	X	X	X	X
Cipher text translation	X	X	X	X
HMAC		X		X
MAC (AES and DES)	X	X	X	X
PIN (AES and DES)	X	X	X	X
Transport keys (AES and DES)	X	X	X	X
Key-generating (AES and DES)	X	X	X	X

Entering keys by using the key generator utility program: One function that KGUP performs is to enter key values that you supply into the CKDS. You can enter a clear or encrypted key value by using KGUP.

You submit KGUP control statements to specify to KGUP the function that you want KGUP to perform. To enter a key, you specify the key value in a KGUP control statement. You can either specify an encrypted or clear key value.

When you enter an encrypted key value, the key value must be encrypted under an importer key-encrypting key that exists in the CKDS. You use the KGUP control statement to specify which importer key-encrypting key encrypts the key. KGUP reenciphers the key from under the importer key-encrypting key to under the master key and places the key in the CKDS.

When you enter a clear key value, KGUP enciphers the clear key value under the master key and places the key in the CKDS. Because entering clear keys may endanger security, ICSF must be in special secure mode before you can enter a clear key by using KGUP. Special secure mode lowers the security of your system to allow you to use KGUP to enter clear keys and to produce clear PINs.

Special secure mode: To use special secure mode, you must either:

- Define the CSF.SSM.ENABLE SAF profile in the XFACILIT SAF resource class.
- Specify YES for the SSM installation option in the installation options data.

  For information about specifying installation options, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

If these conditions permit the use of special secure mode, it is enabled automatically when you specify that you are entering clear key values in a KGUP statement.

For a detailed description of how to use KGUP to enter keys, see Chapter 11, "Managing Cryptographic Keys Using the Key Generator Utility Program," on page 185.

Entering keys by using the dynamic CKDS update services: ICSF provides a set of callable services that allow applications to dynamically update the CKDS. Applications can use the CKDS Key Record Create service to create new records in the CKDS, the CKDS Key Record Write service to write a key token to an existing record, and CKDS Key Record Delete service to remove a record from the CKDS. These dynamic updates affect both the DASD copy of the CKDS currently in use and the in-storage copy. Another service allows an application to retrieve the key token from a record in the in-storage CKDS. That token can be used directly in subsequent CALLs to cryptographic services. The Key Part Import and Key Part Import2 callable services combines the clear key parts and returns the key value either in an internal key token or as a dynamic update to the CKDS. For more information on using the dynamic CKDS update services or the key part import services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Entering keys into the PKDS

All ECC and RSA public and private keys and trusted blocks may be stored in the public key data set (PKDS).

There are several methods you can use to enter keys into the PKDS:

- Callable services

  You can use the PKA Key Generate callable service to update a skeleton token in the PKDS with a generated private key token.
- Dynamic PKDS update callable services

  You can program applications to use the PKDS key record create service to create new entries in the PKDS and the PKDS key record write service to enter key tokens into the PKDS.
- Trusted Key Entry (TKE) workstation

  RSA and ECC private keys can be imported from the TKE workstation and stored in the PKDS. For more information, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.
- Enterprise Key Management Foundation (EKMF)

  The Enterprise Key Management Foundation (EKMF) provides online key management to ICSF as well as to IBM cryptographic products on other platforms. EKMF offers centralized key management for CCA symmetric and asymmetric keys and for certificates. EKMF automates the key management process and exchanges and replaces keys and certificates on demand. Also, to assure continuous operation, EKMF maintains backup copies of all critical keys.

  For additional information, contact the Crypto Competence Center at **ccc@dk.ibm.com** or at: **https://www-304.ibm.com/jct05001c/dk/security/cccc/**.
- ICSF PKDS Key Management panels

  RSA keys in the PKDS can be managed using the PKDS key management panel utilities.
  - You can generate an RSA key which is stored in the PKDS.
  - You can delete any key from the PKDS.
  - You can create an X.509 certificate to export an RSA public key in the PKDS.

– You can import an RSA public key from an X.509 certificate and store it in the PKDS. For more information, see Chapter 15, "Using the utility panels to manage keys in the PKDS," on page 317.

Entering keys by using the dynamic PKDS update services: ICSF provides a set of callable services that allow applications to dynamically update the PKDS. Applications can use the PKDS Key Record Create service to create new records in the PKDS, the PKDS Key Record Write service to write a key token to an existing record, and PKDS Key Record Delete service to remove a record from the PKDS. These dynamic updates affect both the DASD copy of the PKDS currently in use and the in-storage copy. Another service allows an application to retrieve the key token from a record in the in-storage PKDS. That token can be used directly in subsequent CALLs to cryptographic services. For more information on using the dynamic PKDS update services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Maintaining cryptographic keys

CKDS

You can use either KGUP, the dynamic CKDS update services, or Enterprise Key Management Foundation (EKMF) to generate and enter keys into the CKDS or to maintain keys already existing in the CKDS. The keys are stored in records. A record exists for each key that is stored in the CKDS.

A record in the CKDS is called a *key entry* and has a label associated with it. When you call some ICSF callable services, you specify a key label as a parameter to identify the key for the callable service to use.

Use KGUP to change the key value of an entry, rename entry labels, and delete entries in the CKDS. For more information about how to use KGUP to update key entries in the CKDS, see Chapter 11, "Managing Cryptographic Keys Using the Key Generator Utility Program," on page 185.

Use the dynamic CKDS update services in applications to create entries, change the key value of an entry, and delete entries in the CKDS.

You can use SAF to control which applications can use specific keys and services. For more information, see "System authorization facility (SAF) controls" on page 81.

One or more resource profiles in the XFACILIT class define your Key Store Policy. A Key Store Policy consists of a number of controls that collectively determine how encrypted key tokens defined in the CKDS can be accessed and used.

PKDS

You can use either the dynamic PKDS update services, PKDS Key Management panels, the TKE workstation, or Enterprise Key Management Foundation (EKMF) to generate and enter keys into the PKDS or to maintain keys already existing in the PKDS. The keys are stored in records. A record exists for each key that is stored in the PKDS.

A record in the PKDS is called a *key entry* and has a label associated with it. When you call some ICSF callable services, you specify a key label as a parameter to identify the key for the callable service to use.

Use the dynamic PKDS update services in applications to create entries, change the key value of an entry, and delete entries in the PKDS.

You can use SAF to control which applications can use specific keys and services. For more information, see "System authorization facility (SAF) controls" on page 81.

One or more resource profiles in the XFACILIT class define your Key Store Policy. A Key Store Policy consists of a number of controls that collectively determine how encrypted key tokens defined in the PKDS can be accessed and used.

Key Store Policy

A Key Store Policy defines rules for how encrypted key tokens stored in a CKDS or PKDS can be accessed and used. A Key Store Policy is collectively defined by a number of separate controls that each specify a particular rule. Most of the Key Store Policy controls work in conjunction with profiles in the CSFKEYS class and enable you to:

- Specify how ICSF should respond when a key token is passed to a callable service instead of a key label (which is needed to perform a SAF authorization check).
- Determine if applications should be prevented from creating a new key record (with a new key label) for a token that is already stored in the CKDS or PKDS (in a key record with a different key label).
- Specify if READ access authority is sufficient to create, write to, or delete a key label, or if a higher level of access authority should be required for these actions.
- Specify if READ access authority to an AES or DES key is sufficient to export the key (move it from encryption under a master key to encryption under an RSA key), or if UPDATE authority should be required for this action.
- Place restrictions on how keys can be used. You can:
  - Restrict a particular AES or DES key from being exported, or allow it to be exported only by certain RSA keys (or only by RSA keys bound to identities in certain key certificates).
  - Restrict certain RSA keys from being used in secure export and import operations, or from being used in handshake operations.
- Allows archived records in the CKDS and PKDS to be used by applications.
- Allows archived object records in the TKDS to be used by applications.

Each Key Store Policy control is a resource in the XFACILIT class, and can be enabled by creating a profile for the resource using the RDEFINE command. Similarly, you can disable a control by deleting its profile using the RDELETE command.

Certain controls, when enabled, will *activate* Key Store Policy for either the CKDS or PKDS. When Key Store Policy is *activated*, ICSF will identify the key label or labels associated with each key token in the key store. This information is needed, for example, in order to carry out SAF authorization checks against RACF profiles (which are based on key labels) when a key token is passed to a callable service, or to ensure an application does not store a duplicate token (a token that is already stored, but associated with a different key label) in the key store. In addition to the controls that activate Key Store Policy, other controls that do not themselves activate Key Store Policy may still require, or to a lesser degree rely upon, an active Key Store Policy and its key token/label associations. The following table outlines the Key Store Policy controls that are available. This table also highlights the controls that activate Key Store Policy for a CKDS or PKDS, as well as the dependencies the other controls have on Key Store Policy being active. Be aware that Key Store Policy is activated separately for a CKDS and a PKDS.

Defining a Key Store Policy: A Key Store Policy is made up of a number of controls. Each Key Store Policy control is a resource in the XFACILIT class. The existence of a profile for a particular resource in the XFACILIT class enables that control. A Key Store Policy applies only to encrypted keys in a CKDS or PKDS.

Table 25. Key Store Policy controls

The following Key Store Policy controls:	Consist of the following XFACILIT class resources:	Description:
Key Token Authorization Checking controls Verifies, when an application passes a callable service a key token instead of a key label, that the user has authority to the key token in the CKDS or PKDS. It does this by identifying the key label associated with the passed token.	CSF.CKDS.TOKEN.CHECK.LABEL.WARN	**Activates Key Store Policy for CKDS.** Enables Key Token Authorization Checking for the CKDS in warning mode. In this mode, a failing authorization check will result in a warning, but the operation will be allowed to continue.
	CSF.CKDS.TOKEN.CHECK.LABEL.FAIL	**Activates Key Store Policy for CKDS.** Enables Key Token Authorization Checking for the CKDS in fail mode. In this mode, ICSF does not allow the operation to continue when the authorization check fails. The service returns with an error.
	CSF.PKDS.TOKEN.CHECK.LABEL.WARN	**Activates Key Store Policy for PKDS.** Enables Key Token Authorization Checking for the PKDS in warning mode. In this mode, a failing authorization check will result in a warning, but the operation will be allowed to continue.
	CSF.PKDS.TOKEN.CHECK.LABEL.FAIL	**Activates Key Store Policy for PKDS.** Enables Key Token Authorization Checking for the PKDS in fail mode. In this mode, ICSF does not allow the operation to continue when the authorization check fails. The service returns with an error.
Default Key Label Checking controls Specifies that ICSF should use a default profile to determine application access to tokens that are not stored in the CKDS or PKDS. Can be enabled only if the Key Token Authorization Checking control for the appropriate key store is also enabled.	CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL	**Requires an active Key Store Policy for CKDS.** Specifically, this control can be enabled only if the CSF.CKDS.TOKEN.CHECK.LABEL.WARN or CSF.CKDS.TOKEN.CHECK.LABEL.FAIL control is also enabled. Specifies that ICSF should use the default profile CSF-CKDS-DEFAULT in the CSFKEYS class to determine user access to tokens that are not stored in the CKDS.
	CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL	**Requires an active Key Store Policy for PKDS.** Specifically, this control can be enabled only if the CSF.PKDS.TOKEN.CHECK.LABEL.WARN or CSF.PKDS.TOKEN.CHECK.LABEL.FAIL control is also enabled. Specifies that ICSF should use the default profile CSF-PKDS-DEFAULT in the CSFKEYS class to determine user access to tokens that are not stored in the PKDS.

Table 25. Key Store Policy controls (continued)

The following Key Store Policy controls:	**Consist of the following XFACILIT class resources:**	**Description:**
Duplicate Key Token Checking controls Prevents applications from storing duplicate tokens in the CKDS or PKDS.	CSF.CKDS.TOKEN.NODUPLICATES	**Activates Key Store Policy for CKDS.** Enables Duplicate Key Token Checking for the CKDS. ICSF will prevent an application from creating a new key record (with a new key label) for a token that is already stored in the CKDS.
	CSF.PKDS.TOKEN.NODUPLICATES	**Activates Key Store Policy for PKDS.** Enables Duplicate Key Token Checking for the PKDS. ICSF will prevent an application from creating a new key record (with a new key label) for a token that is already stored in the PKDS.
Granular Key Label Access controls Increases the level of access authority required to create, write to, or delete a key label.	CSF.CSFKEYS.AUTHORITY.LEVELS.WARN	Enables Granular Key Label Access in warning mode. In this mode, a warning will be issued if the user does not have UPDATE authority (if creating a label), or CONTROL authority (if writing to or deleting a label). As long as the user has READ authority, however, ICSF will allow the operation to continue. **Does not require an active Key Store Policy for CKDS or PKDS. However, if a key token is passed to a callable service instead of a key label, ICSF will, in order to initiate a SAF authorization check, rely on an active Key Store Policy for the appropriate key store.**
	CSF.CSFKEYS.AUTHORITY.LEVELS.FAIL	Enables Granular Key Label Access in fail mode. In this mode, ICSF will not allow a key label to be modified if the user does not have UPDATE authority (if creating a label), or CONTROL authority (if writing to or deleting a label). The service returns with an error. **Does not require an active Key Store Policy for CKDS or PKDS. However, if a key token is passed to a callable service instead of a key label, ICSF will, in order to initiate a SAF authorization check, rely on an active Key Store Policy for the appropriate key store.**
Symmetric Key Label Export controls Specifies that profiles in the XCSFKEY class (instead of profiles in the CSFKEYS class) should be used to determine access to AES or DES keys that an application is attempting to export using the Symmetric Key Export (CSNDSYX, CSNFSYX, or CSNDSXD) callable service. This allows you to control access to AES and DES keys for the purpose of key export separately from the access allowed to the keys for other purposes.	CSF.XCSFKEY.ENABLE.AES	Enables Symmetric Key Label Export for AES keys. Specifies that profiles in the XCSFKEY class should determine access to an AES key when an application is attempting to export it using the Symmetric Key Export callable services: CSNDSYX or CSNDSXD. **Does not require an active Key Store Policy for CKDS. Note that the Key Token Authorization Checking control must be enabled in order for a SAF authorization check to be performed on key tokens.**

Table 25. Key Store Policy controls (continued)

The following Key Store Policy controls:	Consist of the following XFACILIT class resources:	Description:
	CSF.XCSFKEY.ENABLE.DES	Enables Symmetric Key Label Export for DES keys. Specifies that profiles in the XCSFKEY class should determine access to a DES key when an application is attempting to export it using the Symmetric Key Export callable services: CSNDSYX or CSNDSXD. **Does not require an active Key Store Policy for CKDS. Note that the Key Token Authorization Checking control must be enabled in order for a SAF authorization check to be performed on key tokens.**
PKA Key Management Extensions control Specifies that the ICSF segment of profiles in the CSFKEYS class (and the XCSFKEY class when a Symmetric Key Label Export control is enabled) will be checked to determine additional restrictions on how keys covered by the profile can be used.	CSF.PKAEXTNS.ENABLE.WARNONLY	**Requires an active Key Store Policy for CKDS and PKDS.** Enables PKA Key Management Extensions in warning mode. The ICSF segment of CSFKEYS or XCSFKEY profiles will be checked to: • determine if a symmetric key can be exported, and, if so, which asymmetric keys can be used in the operation to re-encrypt the symmetric key. • determine if an asymmetric key can be used in secure export and import operations, or in handshake operations. However, because this is warning mode, ICSF will allow the operation to continue even if the ICSF segment indicates that the operation is not allowed.
	CSF.PKAEXTNS.ENABLE	**Requires an active Key Store Policy for CKDS and PKDS.** Enables PKA Key Management Extensions in fail mode. The ICSF segment of CSFKEYS or XCSFKEY profiles will be checked to: • Determine if a symmetric key can be exported, and, if so, which asymmetric keys can be used in the operation to re-encrypt the symmetric key. • Determine if an asymmetric key can be used in secure export and import operations, or in handshake operations. If the ICSF segment indicates that the operation is not allowed, the service returns with an error.
Key Archive Use control Specifies that ICSF allows an application to use the key material of a CKDS, PKDS, or TKDS record that has been archived.	CSF.KDS.KEY.ARCHIVE.USE	Enables the Key Archive Use control. ICSF will not fail a service request using the label of an archived CKDS, PKDS, or TKDS record.

For more information on the:

- Key Token Authorization Checking controls, refer to “Enabling access authority checking for key tokens” on page 43.
- Default Key Label Checking controls, refer to “Determining access to tokens not stored in the CKDS or PKDS” on page 44.
- Duplicate Key Token Checking controls, refer to “Enabling duplicate key label checking” on page 45.
- Granular Key Label Access controls, refer to “Increasing the level of authority needed to modify key labels” on page 46.
- Symmetric Key Label Export controls, refer to “Increasing the level of authority required to export symmetric keys” on page 48.

- PKA Key Management Extension control, refer to “Controlling how cryptographic keys can be used” on page 51.
- Key Archive Use controls, refer to “Enabling use of archived KDS records” on page 61.

Enabling access authority checking for key tokens: Profiles in the CSFKEYS class determine access authority to cryptographic keys. However, CSFKEYS profiles protect keys by their key label (discrete or generic CSFKEYS profiles are named to match one or more key labels), and ICSF callable services accept either a key label or key token. By default, if an application passes a callable service a key token instead of a key label, no authorization checking is done on the use of the key. By enabling Key Token Authorization Checking controls, you can have ICSF identify a key token's associated key label so that a SAF authorization check can be performed. This lets you implement a consistent security policy for keys regardless of how they are identified (by key label or key token) to callable services.

Separate Key Token Authorization Checking controls are provided for activating the checking for either a CKDS or a PKDS in either warning or fail mode. In warning mode, authorization checking is performed, but an application will not be prevented from using a token even when the user lacks the necessary authority. Instead, ICSF will merely log an SMF type 82 subtype 25 record in the SMF data set. Warning mode allows you to identify users who will need access permission to a key prior to moving to a stricter implementation of the Key Token Authorization Checking policy.

This stricter implementation of the policy is called fail mode. In fail mode, an application will be denied access to a token when the user does not have authority to access it. The operation will be unsuccessful, and a return code 8, reason code BF7 (3063) will be returned to the calling application. As with warning mode, ICSF will log an SMF type 82 subtype 25 record in the SMF data set. In addition, RACF will log an SMF type 80 record (with event code qualifier of ACCESS). The resource name in the SMF type 80 record will be the first label associated with the key token that failed the check.

Because the same token could be associated with multiple key records in the key store, when an application passes an encrypted key token to an ICSF callable service, ICSF locates all the labels associated with the passed token. If the user has permission to any of the key labels, then the application is granted authority to use the token. Because access authority to any label associated with a token will give a user access to the token, you may want to ensure that the key store does not contain multiple key records for the same key token. ICSF provides a utility program, CSFDUTIL, that generates a report of all duplicate keys for either a CKDS or PKDS. To prevent duplicate keys from being added to a key store, you can enable the Duplicate Key Token Checking control for either the CKDS or PKDS as described in “Enabling duplicate key label checking” on page 45.

If ICSF cannot find an associated key label for the passed token in the key store, no authorization checking will be performed on the use of the key unless the Default Key Label Checking control is enabled for the key store. If the Default Key Label Checking control is enabled (as described in “Determining access to tokens not stored in the CKDS or PKDS” on page 44), a default profile will determine user access when ICSF cannot identify an associated label for the passed token.

The following table shows the controls for enabling Key Token Authorization Checking for the CKDS and PKDS in either warning or fail mode. To enable one of the Key Token Authorization Checking controls, create the appropriate profile in the XFACILIT class.

Table 26. Key Store Policy controls: The Key Token Authorization Checking controls

The existence of this resource profile in the XFACILIT class:	Does this:
CSF.CKDS.TOKEN.CHECK.LABEL.WARN	**Activates Key Store Policy for CKDS.** Enables Key Token Authorization Checking for the CKDS in warning mode. In this mode, a failing authorization check will result in a warning, but the operation will be allowed to continue.
CSF.CKDS.TOKEN.CHECK.LABEL.FAIL	**Activates Key Store Policy for CKDS.** Enables Key Token Authorization Checking for the CKDS in fail mode. In this mode, ICSF does not allow the operation to continue when the authorization check fails. The service returns with an error.
CSF.PKDS.TOKEN.CHECK.LABEL.WARN	**Activates Key Store Policy for PKDS.** Enables Key Token Authorization Checking for the PKDS in warning mode. In this mode, a failing authorization check will result in a warning, but the operation will be allowed to continue.
CSF.PKDS.TOKEN.CHECK.LABEL.FAIL	**Activates Key Store Policy for PKDS.** Enables Key Token Authorization Checking for the PKDS in fail mode. In this mode, ICSF does not allow the operation to continue when the authorization check fails. The service returns with an error.

For example, say you want to enable Key Token Authorization Checking for both a CKDS and a PKDS. You're not certain all the users currently accessing key tokens in these key stores will have the necessary access authority, and do not want to disrupt current work patterns at your installation. For this reason, you decide to allow a warning period during which you can identify users who will need permission to access certain key tokens. The following commands will enable Key Token Authorization Checking for the CKDS and the PKDS in warning mode.

```
RDEFINE XFACILIT CSF.CKDS.TOKEN.CHECK.LABEL.WARN
RDEFINE XFACILIT CSF.PKDS.TOKEN.CHECK.LABEL.WARN
SETROPTS RACLIST(XFACILIT) REFRESH
```

During the warning period, you can, by examining the SMF type 82 subtype 25 records logged in the SMF data set, identify the users who need permission to access keys. You can then create or modify the necessary profiles in the CSFKEYS class. When you are ready to move to a stricter implementation of this policy, you enable the controls for fail mode and disable the ones for warning mode.

```
RDEFINE XFACILIT CSF.CKDS.TOKEN.CHECK.LABEL.FAIL
RDEFINE XFACILIT CSF.PKDS.TOKEN.CHECK.LABEL.FAIL
RDELETE XFACILIT CSF.CKDS.TOKEN.CHECK.LABEL.WARN
RDELETE XFACILIT CSF.PKDS.TOKEN.CHECK.LABEL.WARN
SETROPTS RACLIST(XFACILIT) REFRESH
```

If you accidentally enable the Key Token Authorization Checking controls for both warning and fail mode, the control for fail mode will take precedence.

Determining access to tokens not stored in the CKDS or PKDS: When the Key Token Authorization Checking control for a key store has been enabled and a token is passed to a callable service, ICSF will find the key label or labels associated with the passed token so that a SAF authority check can be performed. If, however, the token passed to the callable service is not in the key store, there will be no associated key label to find. By default, no authorization checking is performed on the use of the key, and the operation is allowed. If you enable the

Default Key Label Checking control for the CKDS or PKDS, however, ICSF will use a default profile to determine user access to tokens that are not in the key store.

Separate controls are provided for enabling Default Key Label Checking for a CKDS or a PKDS, The Default Key Label Checking control will be enabled only if the Key Token Authorization Checking control for the appropriate key store is also enabled. Refer to "Enabling access authority checking for key tokens" on page 43 for more information. To enable one the Default Key Label Checking controls, create the appropriate profile in the XFACILIT class.

Table 27. Key Store Policy controls: The Default Key Label Checking controls

The existence of this resource profile in the XFACILIT class:	Does this:
CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL	Specifies that ICSF should use the default profile CSF-CKDS-DEFAULT in the CSFKEYS class to determine user access to tokens that are not stored in the CKDS. This control is enabled only if the CSF.CKDS.TOKEN.CHECK.LABEL.WARN or CSF.CKDS.TOKEN.CHECK.LABEL.FAIL control is also enabled.
CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL	Specifies that ICSF should use the default profile CSF-PKDS-DEFAULT in the CSFKEYS class to determine user access to tokens that are not stored in the PKDS. This control is enabled only if the CSF.PKDS.TOKEN.CHECK.LABEL.WARN or CSF.PKDS.TOKEN.CHECK.LABEL.FAIL control is also enabled.

For example, to enable the Default Key Label Checking control for a CKDS, you would:

1. Create the default profile CSF-CKDS-DEFAULT in the CSFKEYS class.
   ```
   RDEFINE CSFKEYS CSF-CKDS-DEFAULT UACC(NONE)
   ```
2. By defining the universal access authority (UACC) as NONE in the preceding step, the use of key tokens that do not reside in the key store has been prohibited. If necessary, however, you can give appropriate users (preferably groups) access in the CSF-CKDS-DEFAULT profile and refresh the CSFKEYS class in storage:
   ```
   PERMIT CSF-CKDS-DEFAULT CLASS(CSFKEYS) ID(group-id) ACCESS(READ)
   SETROPTS RACLIST(CSFKEYS) REFRESH
   ```
3. Create a profile for the CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL resource in the XFACILIT class, and refresh the XFACILIT class in storage.
   ```
   RDEFINE XFACILIT CSF.CKDS.TOKEN.CHECK.DEFAULT.LABEL
   SETROPTS RACLIST(XFACILIT) REFRESH
   ```

Enabling duplicate key label checking: A key token could be stored in a key store within multiple key records and so could be associated with multiple key labels. When the Key Token Authorization Checking control is enabled for the key store, duplicate tokens can cause problems because all labels that are associated with a key token passed to an ICSF callable service will be used to determine user access to that token. Although you may deliberately restrict access to a token by one of the labels associated with it, a user might still have access to the token through another label. You can enable the Duplicate Key Token Checking control for the CKDS or PKDS to prevent applications from storing duplicate tokens in the key store. When enabled, ICSF services that update the key store will check for duplicate tokens. ICSF will not allow a key token to be written to the key store if it matches a token that is already stored. The Duplicate Key Token Checking controls do not rely on SAF authorization checks against CSFKEYS class profiles. Instead, the callable services that update the key store will verify that a duplicate token does not already exist within the key store.

Note: Enabling the Duplicate Key Token Checking control for the CKDS or PKDS ensures only that no duplicate keys are added to the key store. To identify any duplicate key tokens that may already exist in a CKDS or PKDS, use the CSFDUTIL utility program. The CSFDUTIL utility program generates a report of all duplicate keys in either a CKDS or a PKDS.

Separate controls are provided for enabling Duplicate Key Token Checking for a CKDS or a PKDS. To enable either of the Duplicate Key Token Checking controls, create the appropriate profile in the XFACILIT class.

Table 28. Key Store Policy controls: The Duplicate Key Token Checking controls

The existence of this resource profile in the XFACILIT class:	Does this:
CSF.CKDS.TOKEN.NODUPLICATES	**Activates Key Store Policy for CKDS.** Enables Duplicate Key Token Checking for the CKDS. ICSF will prevent an application from creating a new key record (with a new key label) for a token that is already stored in the CKDS.
CSF.PKDS.TOKEN.NODUPLICATES	**Activates Key Store Policy for PKDS.** Enables Duplicate Key Token Checking for the PKDS. ICSF will prevent an application from creating a new key record (with a new key label) for a token that is already stored in the PKDS.

For example, to ensure that duplicate tokens are not stored in either the CKDS or PKDS, you would enter the following commands:

```
RDEFINE XFACILIT CSF.CKDS.TOKEN.NODUPLICATES
RDEFINE XFACILIT CSF.PKDS.TOKEN.NODUPLICATES
SETROPTS RACLIST(XFACILIT) REFRESH
```

Increasing the level of authority needed to modify key labels: A number of ICSF callable services enable an application to create, write to, or delete a key label. By default, the user needs only READ authority to read from, create, write to, or delete a label. In some cases, however, you might want to require a higher level of authority for modifying a label than is required to merely read a label. By enabling the Granular Key Label Access control, you increase the level of access authority required to create, write to, or delete a label, while still requiring only READ authority for cryptographic functions. This way, you can give a user permission to access a key for encryption or decryption operations, while preventing that same user from changing or deleting the key record.

The following table outlines the increased access authority required when the Granular Key Label Access control is enabled.

Table 29. Increased access authority required to modify key labels when Granular Key Label Access control is enabled

To do this:	The level of access authority required is increased from READ to:	This impacts the following callable services:
Create a label	UPDATE	Key Record Create / Key Record Create2 PKDS Record Create

Table 29. Increased access authority required to modify key labels when Granular Key Label Access control is enabled (continued)

To do this:	The level of access authority required is increased from READ to:	This impacts the following callable services:
Write to a label	CONTROL	Key Part Import / Key Part Import2 Key Record Create / Key Record Create2 Key Record Write / Key Record Write2 PKDS Record Create PKDS Record Write PKA Key Generate PKA Key Import Trusted Block Create
Delete a label	CONTROL	Key Record Delete PKDS Record Delete Retained Key Delete

You can enable the Granular Key Label Access control in warning or fail mode. In warning mode, the user's access authority will be checked, but only READ authority will be required. However, if a user does not have UPDATE authority when creating a label, or CONTROL authority when writing to or deleting a label, a warning will be issued and the access will be logged. Warning mode allows you to identify any users who will need to be granted increased access authority prior to moving to a stricter implementation of the policy. The stricter implementation of the policy is called fail mode. In fail mode, users who lack the increased access authority required will not be able to modify key labels. The operation will be unsuccessful, and a return code of 8 (reason code 16004) will be returned to the calling application.

It is recommended that you activate Key Store Policy for both the CKDS and the PKDS before enabling the Granular Key Label Access control. If Key Store Policy is not activated and the Granular Key Label Access control is enabled, the increased access authority checks will work only when the application passes a callable service a key label. However, if the application were to pass the callable service a key token instead of a key label, then no authorization checking will be performed. When a token is passed, ICSF will, in order to initiate a SAF authorization check, rely on an active Key Store Policy for the appropriate key store.

Enabling any one of the following controls will activate Key Store Policy for a CKDS:

- CSF.CKDS.TOKEN.CHECK.LABEL.WARN
- CSF.CKDS.TOKEN.CHECK.LABEL.FAIL
- CSF.CKDS.TOKEN.NODUPLICATES

Enabling any one of the following controls will activate Key Store Policy for a PKDS:

- CSF.PKDS.TOKEN.CHECK.LABEL.WARN
- CSF.PKDS.TOKEN.CHECK.LABEL.FAIL

- CSF.PKDS.TOKEN.NODUPLICATES

The following table shows the controls for enabling Granular Key Label Access in warning or fail mode. To enable one of the controls, create the appropriate profile in the XFACILIT class.

Table 30. Key Store Policy controls: The Granular Key Label Access controls

The existence of this resource profile in the XFACILIT class:	Does this:
CSF.CSFKEYS.AUTHORITY.LEVELS.WARN	Enables Granular Key Label Access in warning mode. In this mode, a warning will be issued if the user does not have UPDATE authority if creating a label, or CONTROL authority if writing to or deleting a label. As long as the user has READ authority, however, ICSF will allow the operation to continue.
CSF.CSFKEYS.AUTHORITY.LEVELS.FAIL	Enables Granular Key Label Access in fail mode. In this mode, ICSF will not allow a key label to be modified if the user does not have UPDATE authority if creating a label, or CONTROL authority if writing to or deleting a label. The service returns with an error.

For example, you want to require UPDATE authority to create a label, and CONTROL authority to write to or delete a label. You're not certain all the users currently modifying key labels will have the necessary access authority, and do not want to disrupt current work patterns at your installation. For this reason, you decide to allow a warning period during which you can identify which users will need to be granted increased authority. To do this, you would:

1. Enable the Granular Key Label Access control in warning mode.
   ```
   RDEFINE XFACILIT CSF.CSFKEYS.AUTHORITY.LEVELS.WARN
   SETROPTS RACLIST(XFACILIT) REFRESH
   ```
2. Because you have enabled the control in warning mode, a failing access check will still allow a user to modify the key record (as long as the user has READ authority), but will issue a warning and log the access. Using this information, you can update the appropriate profiles in the CSFKEYS class to grant increased access authority to the appropriate users. For example, if user RITA needs to be able to generate RSA key tokens (by way of the CSNDKRC and CSNDPKG callable services), she will need CONTROL access to the label:
   ```
   PERMIT RITA.RSA.TEST.* CLASS(CSFKEYS) ID(RITA) ACCESS(CONTROL)
   ```
3. When you are ready to move to a stricter implementation of the policy, you would enable the control for fail mode and disable the one for warning mode.
   ```
   RDEFINE XFACILIT CSF.CKDS.TOKEN.CHECK.LABEL.FAIL
   RDELETE XFACILIT CSF.CKDS.TOKEN.CHECK.LABEL.WARN
   SETROPTS RACLIST(XFACILIT) REFRESH
   ```

If you accidentally enable the Granular Key Label Access controls for both warning and fail mode, the control for fail mode will take precedence.

Increasing the level of authority required to export symmetric keys: Using the Symmetric Key Export (CSNDSYX, CSNFSYX, or CSNDSXD) callable service, an application can transfer a symmetric (AES or DES) key from encryption under a master key to encryption under an application-supplied RSA public key. This callable service is used because a secure key (which is encrypted under a master key in the ICSF environment) might need to be shared with a partner, and to transfer it to that partner securely, it will need to be encrypted under an RSA key provided by the partner. The partner will then be able to decrypt it using a corresponding private key.

The export operation performed by the Symmetric Key Export callable service does not fit into a traditional access control hierarchy. Due to the nature of the export operation, you might want to restrict users from accessing a symmetric key for the purpose of exporting it, while still allowing users to access the key for other purposes. By enabling the Symmetric Key Label Export control for AES or DES keys, and creating profiles in the XCSFKEY resource class, you can increase the level of access authority needed to export AES or DES keys without increasing the level of authority needed to access the keys for other operations.

By default, the CSFKEYS class determines access authority to cryptographic keys passed to callable services (including the Symmetric Key Export and Symmetric Key Export with Data services). When the Symmetric Key Label Export control for AES or DES keys is **not** enabled and the Symmetric Key Export or Symmetric Key Export with Data service is called, a user needs only READ authority for the key (as specified in a CSFKEYS class profile). If, however, the Symmetric Key Label Export control for AES or DES keys **is** enabled and the Symmetric Key Export or Symmetric Key Export with Data callable service is called, then a user needs UPDATE authority for the key (as specified in an XCSFKEY class profile). The Symmetric Key Label Export controls affect only the Symmetric Key Export and Symmetric Key Export with Data services; for all other callable services, access to cryptographic keys is checked against profiles in the CSFKEYS class. Furthermore, the Symmetric Key Label Export controls affect access only to the symmetric key the application is attempting to export and do not affect access to the RSA key that is being used to re-encrypt the symmetric key. Access authority to the AES or DES key will be checked against XCSFKEY class profiles, while access to the RSA key will still be checked against CSFKEYS class profiles.

It is recommended that you activate Key Store Policy for the CKDS before enabling the Symmetric Key Label Export control for AES or DES keys. If Key Store Policy is not activated for the CKDS and the Symmetric Key Label Export control for AES or DES keys is enabled, the access authority check for the symmetric key will be performed only when it is identified to the Symmetric Key Export and Symmetric Key Export with Data callable services by its key label. If the application were to pass the callable service a key token instead of a key label, then no authorization checking will be performed. When a token is passed, ICSF will, in order to initiate a SAF authorization check, rely on an active Key Store Policy for CKDS. Enabling any one of the following controls will activate Key Store Policy for a CKDS:

- CSF.CKDS.TOKEN.CHECK.LABEL.WARN
- CSF.CKDS.TOKEN.CHECK.LABEL.FAIL
- CSF.CKDS.TOKEN.NODUPLICATES

The following table shows the controls for enabling Symmetric Key Label Export for AES or DES keys. To enable the controls, create the appropriate profile in the XFACILIT class. There are separate Symmetric Key Label Export controls for AES and DES keys so you can require UPDATE authority (which will be checked against XCSFKEY profiles) for export of one type of key, while still requiring only READ authority (which will still be checked against CSFKEY profiles) for export of the other type of key. There are no Symmetric Key Label Export controls that enable the policy in a warning mode. However, you can use the WARNING operand on XCSFKEY profiles to achieve the same results.

Table 31. Key Store Policy controls: The Symmetric Key Label Export controls

The existence of this resource profile in the XFACILIT class:	Does this:
CSF.XCSFKEY.ENABLE.AES	Enables Symmetric Key Label Export for AES keys. Specifies that profiles in the XCSFKEY class should determine access to an AES key when an application is attempting to export it using the Symmetric Key Export (CSNDSYX, CSNFSYX, or CSNDSXD) callable service.
CSF.XCSFKEY.ENABLE.DES	Enables Symmetric Key Label Export for DES keys. Specifies that profiles in the XCSFKEY class should determine access to a DES key when an application is attempting to export it using the Symmetric Key Export (CSNDSYX, CSNFSYX, or CSNDSXD) callable service.

For example, you want to require UPDATE authority to export any symmetric key (AES or DES) using the Symmetric Key Export callable service. You're not certain all the users currently exporting symmetric keys will have the necessary access authority, and do not want to disrupt current work patterns at your installation. For this reason, you decide to allow a warning period during which you can identify which users will need to be granted increased authority. To do this, you would:

1. Create profiles in the XCSFKEY class to cover the symmetric keys. In this example, your installation has a consistent naming policy for AES and DES key labels, so the following two generic profiles will cover all symmetric keys. The WARNING operand is specified to initiate the warning period.

   ```
   RDEFINE XCSFKEY AES* UACC(NONE) WARNING
   RDEFINE XCSFKEY DES* UACC(NONE) WARNING
   ```

   The XCSFKEY class will need to be activated and placed in common storage:

   ```
   SETROPTS CLASSACT(XCSFKEY)
   SETROPTS RACLIST(XCSFKEY)
   ```

2. Enable the Symmetric Key Label Export control for AES and DES. In this example, we enable both controls so that UPDATE authority is required when exporting any symmetric key.

   ```
   RDEFINE XFACILIT CSF.XCSFKEY.ENABLE.AES
   RDEFINE XFACILIT CSF.XCSFKEY.ENABLE.DES
   ```

3. Because the WARNING operand was specified on the generic profiles AES* and DES*, any failing access check will still allow access to the symmetric key, but will issue a warning message and log the access. Using this information, you can grant UPDATE access to users or groups as needed. Since the generic profiles in our example cover all AES and all DES keys, you may need to create other generic profiles or discrete profiles to limit access for certain users. Here, user BOBADMIN is given UPDATE access to all symmetric keys, while user GWEN is given UPDATE access to the key labeled DES.BURDA.MEDINC.

   ```
   PERMIT AES* CLASS(XCSFKEY) ID(BOBADMIN) ACCESS(UPDATE)
   PERMIT DES* CLASS(XCSFKEY) ID(BOBADMIN) ACCESS(UPDATE)
   RDEFINE XCSFKEY DES.BURDA.MEDINC UACC(NONE)
   PERMIT DES.BURDA.MEDINC CLASS(XCSFKEY) ID(GWEN) ACCESS(UPDATE)
   ```

   The XCSFKEY class will need to be refreshed in common storage:

   ```
   SETROPTS RACLIST(XCSFKEY) REFRESH
   ```

4. When you are ready to move to a stricter implementation of the policy, you can end the warning period. To do this, update the necessary profiles in the XCSFKEY class using the RALTER command with its NOWARNING operand.

   ```
   RALTER XCSFKEY AES* UACC(NONE) NOWARNING
   RALTER XCSFKEY DES* UACC(NONE) NOWARNING
   ```

The XCSFKEY class will need to be refreshed in common storage:

```
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Controlling how cryptographic keys can be used: In addition to using profiles in the CSFKEYS class (and, when Symmetric Key Label Export is enabled, the XCSFKEY class) to identify which users have permission to certain cryptographic keys, you can also enable the PKA Key Management Extensions control so that CSFKEYS and XCSFKEY profiles can place restrictions on how keys are used. For example, you can:

- Restrict an asymmetric key from being used in secure export and import operations.
- Restrict an asymmetric key from being used in handshake operations.
- Restrict a symmetric key from being exported (transferred from encryption under a master key to encryption under an application-supplied RSA public key). Alternatively, you can allow the symmetric key to be exported, but only by certain public keys (as indicated by a list of key labels), or only by public keys bound to certain identities (as indicated by a list of certificates in either a PKCS #11 token, or a SAF key ring).

Setting restrictions such as these can help ensure that keys are used only for intended purposes, regardless of who has access to the keys. For example, if you have an RSA key pair intended only for generating and verifying digital signatures, you can set a restriction to ensure that the public key of this key pair is never used to export a symmetric key.

You place restrictions on cryptographic keys using the ICSF segment of the CSFKEYS or XCSFKEY class profiles that cover the keys. After you have modified the profiles with the restrictions you want to place on the keys, you can enable the PKA Key Management Extensions control by creating a CSF.PKAEXTNS.ENABLE profile in class XFACILIT. You can also enable PKA Key Management Extensions in warning mode by creating a CSF.PKAEXTNS.ENABLE.WARNONLY profile in class XFACILIT. In order to enable PKA Key Management Extensions, Key Store Policy must be active for both the CKDS and the PKDS. For more information, refer to "Enabling PKA key management extensions" on page 58.

Restricting asymmetric keys from being used in secure import and export operations: Using the ASYMUSAGE field in the ICSF segment of CSFKEYS profiles enables you to restrict asymmetric keys covered by the profile from being used in secure import and export operations. In secure export operations, a symmetric key (AES or DES) is moved from encryption under a master key to encryption under an asymmetric key (RSA public key). In a secure import operation, the private key of an RSA key pair is used to move a symmetric key from encryption under the RSA public key to encryption under a master key. The following callable services all identify an asymmetric key (either the public or private key of an RSA key pair) to encrypt or decrypt a symmetric key. The callable services that perform secure import and export operations are:

- Symmetric Key Generate (CSNDSYG and CSNFSYG)
- Symmetric Key Export (CSNDSYX, CSNFSYX, CSNDSXD)
- Symmetric Key Import (CSNDSYI and CSNFSYI) and Symmetric Key Import2 (CSNDSYI2 and CSNFSYI2)

For each of these services, a profile in the CSFKEYS class will control access to the asymmetric key. In addition to specifying user access to the key, the CSFKEYS profile can also specify information (in the ICSF segment of the profile) on how the key can be used. The ASYMUSAGE field of the ICSF segment enables you to specify whether an asymmetric key covered by the profile can participate in secure import or export operations. By specifying the NOSECUREEXPORT keyword in the ASYMUSAGE field, you restrict any asymmetric key covered by the profile from being used to encrypt or decrypt the symmetric key in these operations.

For example, the profile RSA.SAMMY.DIGSIG in class CSFKEYS covers an RSA key pair that should be used only for generating and verifying digital signatures and performing TLS/SSL handshakes. The following RALTER command modifies the profile to ensure that the public key of the RSA key pair is never used to export keys. The SETROPTS RACLIST command is used to refresh the profile in common storage.

```
RALTER CSFKEYS RSA.SAMMY.DIGSIG ICSF(ASYMUSAGE(NOSECUREEXPORT))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

In order for the secure import/export restriction to take effect, you will need to enable the PKA Key Management Extensions control by creating a CSF.PKAEXTNS.ENABLE profile in class XFACILIT. In order to enable the PKA Key Management Extensions control, the Key Store Policy for both the CKDS and the PKDS must also be active. Refer to “Enabling PKA key management extensions” on page 58 for more information.

When the PKA Key Management Extensions control is enabled, the default is to allow keys to participate in secure import and export operations. You can also explicitly specify this using the SECUREEXPORT keyword in the ASYMUSAGE field of a CSFKEYS profile. For example:

```
RALTER CSFKEYS RSA.SAMMY.EXPORT ICSF(ASYMUSAGE(SECUREEXPORT))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

The ASYMUSAGE field can also contain the NOHANDSHAKE or HANDSHAKE keywords to specify whether keys covered by the profile can participate in handshake operations (as described in “Restricting asymmetric keys from being used in handshake operations”). These keywords can be specified along with the NOSECUREEXPORT or SECUREEXPORT keywords when entering the RDEFINE or RALTER command.

```
RALTER CSFKEYS RSA.SAMMY.EXPORT ICSF(ASYMUSAGE(SECUREEXPORT NOHANDSHAKE))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

Restricting asymmetric keys from being used in handshake operations: Using the ASYMUSAGE field in the ICSF segment of CSFKEYS profiles enables you to restrict asymmetric keys covered by the profile from being used in handshake operations. The following callable services all identify an asymmetric key to be used in a handshake operation. The callable services that perform handshake operations are:

- Digital Signature Generate (CSNDDSG and CSNFDSG)
- Digital Signature Verify (CSNDDSV and CSNFDSV)
- PKA Encrypt (CSNDPKE and CSNFPKE)
- PKA Decrypt (CSNDPKD and CSNFPKD)

For each of these services, a profile in the CSFKEYS class will control access to the asymmetric key used to generate/verify a digital signature, or encrypt/decrypt a clear key value. In addition to specifying user access to the key, the CSFKEYS profile can also specify information (in the ICSF segment of the profile) on how the key can be used. The ASYMUSAGE field of the ICSF segment enables you to specify whether an asymmetric key covered by the profile can participate in handshake operations. By specifying the NOHANDSHAKE keyword in the

ASYMUSAGE field, you restrict any key covered by the profile from being used in handshake operations. For example, the profile RSA.SAMMY.EXPORT in class CSFKEYS covers an RSA key pair intended for exporting and importing symmetric keys. The following RALTER command modifies the profile to ensure that the RSA keys are not used in handshake operations. The SETROPTS RACLIST command is used to refresh the profile in common storage.

```
RALTER CSFKEYS RSA.SAMMY.EXPORT ICSF(ASYMUSAGE(NOHANDSHAKE))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

In order for the restriction on handshake operations to take effect, you will need to enable the PKA Key Management Extensions control by creating a CSF.PKAEXTNS.ENABLE profile in class XFACILIT. In order to enable the PKA Key Management Extensions control, the Key Store Policy for both the CKDS and the PKDS must also be active. Refer to "Enabling PKA key management extensions" on page 58 for more information.

When the PKA Key Management Extensions control is enabled, the default is to allow keys to participate in handshake operations. You can also explicitly specify this using the HANDSHAKE keyword in the ASYMUSAGE field of profiles in the CSFKEYS class. For example:

```
RALTER CSFKEYS RSA.SAMMY.EXPORT ICSF(ASYMUSAGE(HANDSHAKE))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

The ASYMUSAGE field can also contain the NOSECUREEXPORT or SECUREEXPORT keywords to specify whether keys covered by the profile can participate in secure import and export operations (as described in "Restricting asymmetric keys from being used in secure import and export operations" on page 51). These keywords can be specified along with the NOHANDSHAKE or HANDSHAKE keywords when entering the RDEFINE or RALTER command.

```
RALTER CSFKEYS RSA.SAMMY.EXPORT ICSF(ASYMUSAGE(NOSECUREEXPORT HANDSHAKE))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

Placing restrictions on exporting symmetric keys: The Symmetric Key Export and Symmetric Key Export with Data callable services lets a calling application transfer a symmetric (AES or DES) key from encryption under a master key to encryption under an application-supplied RSA public key. This callable service is needed because a secure key (which is encrypted under a master key in the ICSF environment) might need to be shared with a partner, and to transfer it to that partner securely, it will need to be encrypted under an RSA key provided by the partner. The partner will then be able to decrypt it using a corresponding private key. Due to the nature of the operation performed by the Symmetric Key Export callable service, you may want to place additional restrictions on its use. "Increasing the level of authority required to export symmetric keys" on page 48 describes how you can enable the Symmetric Key Label Export controls to specify that a user needs UPDATE authority in the XCSFKEY class (instead of the default READ authority in the CSFKEYS class) to export a symmetric key. By enabling the PKA Key Management Extensions control, can also specify that a symmetric key covered by a CSFKEYS or XCSFKEY profile:

- Cannot be exported.
- Can be exported by any asymmetric key in the PKDS.
- Can be exported only by certain asymmetric keys in the PKDS (as specified by a supplied list).
- Can be exported by any asymmetric key, provided it is bound to an identity in a key certificate in a trusted certificate repository (either a PKCS #11 token or a SAF key ring).

- Can be exported only by an asymmetric key that is bound to certain identities (as specified by a supplied list of key certificates in a trusted certificate repository).

When an application calls the Symmetric Key Export or Symmetric Key Export with Data service, access to the symmetric key (the AES or DES key to be re-encrypted) is determined by a profile in the CSFKEYS class or, if the Symmetric Key Label Export control has been enabled, the XCSFKEY class. In addition to specifying user access to the key, the CSFKEYS or XCSFKEY profile can also place restrictions (in the ICSF segment of the profile) on export of the symmetric key. In the ICSF segment of a CSFKEYS or XCSFKEY profile, the SYMEXPORTABLE field contains a keyword that determines if the key can be exported, and if so, how ICSF will determine the asymmetric keys (the RSA public keys) that can export (re-encrypt) the key.

Table 32. Keyword settings for symmetric key export using the ICSF segment's SYMEXPORTABLE field

This field/keyword	Specifies:
SYMEXPORTABLE(BYNONE)	The symmetric key cannot be exported.
SYMEXPORTABLE(BYLIST)	The symmetric key can be exported, but only by certain RSA public keys in the PKDS (as specified by a supplied list), or only by RSA public keys bound to certain identities (as specified by a supplied list of key certificates). • To supply a list of RSA public keys in the PKDS that can export the symmetric key, you use the SYMEXPORTKEYS field on the ICSF segment. You can list the RSA public keys by label, or you can use a special character setting in this field to specify that any RSA public key in the PKDS can export the symmetric key. • To supply a list a key certificates, you use the SYMEXPORTCERTS field of the ICSF segment. You can list the certificates by label, or you can use a special character setting in this field to specify that any RSA public key bound to an identity in any certificate in the repository can export the symmetric key.
SYMEXPORTABLE(BYANY)	There are no additional restrictions placed on export of the key. Provided no other access requirement or control prevents it, the symmetric key can be exported by any asymmetric key. This is the default.

- For more information on the BYNONE keyword, refer to "Restricting the symmetric key from being exported."
- For more information on using the BYLIST keyword and the SYMEXPORTKEYS field, refer to "Identifying RSA public keys that can export the symmetric key" on page 55.
- For more information on using the BYLIST keyword and the SYMEXPORTCERTS field, refer to "Identifying key certificates for symmetric key export" on page 56.
- For more information on the BYANY keyword, refer to "Placing no additional restrictions on symmetric key export" on page 58.

Restricting the symmetric key from being exported

CSFKEYS and XCSFKEY profiles can contain an ICSF segment. Fields of the ICSF segment specify rules for key use. In the SYMEXPORTABLE field of the ICSF segment, the BYNONE keyword specifies that the symmetric key or keys covered by the profile cannot be exported regardless of a user's access authority to the key. If an application attempts to use the Symmetric Key Export or Symmetric Key Export with Data service to transfer a symmetric (AES or DES) key covered by the profile, the operation will fail and the service will return an error.

For example, the CKDS contains a DES key labeled DES.BRADY.CASTLE that should never be exported. The Symmetric Key Label Export control for DES keys has not been enabled so the key is covered by a profile in the CSFKEYS class. The following RALTER command modifies the discrete profile DES.BRADY.CASTLE to indicate that the key should never be exported. The SETROPTS RACLIST command is used to refresh the profile in common storage.

```
RALTER CSFKEYS DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYNONE))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

Identifying RSA public keys that can export the symmetric key

CSFKEYS and XCSFKEY profiles can contain an ICSF segment. Fields of the ICSF segment specify rules for key use. In the SYMEXPORTABLE field of the ICSF segment, the BYLIST keyword specifies that the symmetric key or keys covered by the profile can be exported by keys identified using the SYMEXPORTKEYS or SYMEXPORTCERTS fields.

Using the SYMEXPORTKEYS field, you can list the RSA public keys in the PKDS that are allowed to export the symmetric key. The SYMEXPORTKEYS list consists of one or more PKDS key labels identifying the RSA public keys under which the symmetric key can be re-encrypted. These labels follow the normal ICSF label conventions; they can be space separated, and quotes are optional.

Note: Key Store Policy must be active in order for the PKA Key Management Extensions to be enabled. Because Key Store Policy for the PKDS is active, ICSF knows the key label or labels associated with each key token. Tokens associated with multiple labels are considered equivalent. Be aware that as long as one of the labels associated with the token appears in the SYMEXPORTKEYS list, the RSA public key can export symmetric key.

A special key label is the asterisk character (*). If the SYMEXPORTKEYS field contains this special key label, any RSA public key in the PKDS can export the symmetric key (provided no other access requirement or control prevents it).

If an application attempts to use the Symmetric Key Export or Symmetric Key Export with Data callable service to transfer a symmetric (AES or DES) key covered by the profile, ICSF will compare the RSA public key identified by the application with those identified in the SYMEXPORTKEYS list. If the key is in the list, the operation is allowed to continue. If it is not in the list, and is also not bound to an identity in a certificate listed in the SYMEXPORTCERTS field (as described in "Identifying key certificates for symmetric key export" on page 56), the operation will fail and the service will return an error.

For example, the following RALTER command modifies the discrete profile DES.BRADY.CASTLE so that the DES key it covers can be exported only by the RSA public key RSA.BRADY.CASTLE. In this example, the Symmetric Key Label Export control has been enabled for DES keys, so the DES.BRADY.CASTLE profile is defined in the XCSFKEY class. The SETROPTS RACLIST command is used to refresh the profile in common storage.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYLIST) SYMEXPORTKEYS(RSA.BRADY.CASTLE))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

To instead allow any RSA public key in the PKDS to export the symmetric key covered by the DES.BRADY.CASTLE profile, you would specify the asterisk character (*) in the SYMEXPORTKEYS field.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYLIST) SYMEXPORTKEYS(*))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

The ADDSYMEXPORTKEYS keyword of the ICSF segment enables you to add labels to a SYMEXPORTKEYS list without having to re-create the entire list. For example, to add the label RSA.BKNIGHT.CASTLE to the list, you would enter:

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(ADDSYMEXPORTKEYS(RSA.BKNIGHT.CASTLE))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Similarly, you can delete labels from a SYMEXPORTKEYS list using the DELSYMEXPORTKEYS keyword:

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(DELSYMEXPORTKEYS(RSA.BKNIGHT.CASTLE))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

You can also delete the entire SYMEXPORTKEYS field using the NOSYMEXPORTKEYS keyword.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(NOSYMEXPORTKEYS)
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Identifying key certificates for symmetric key export

CSFKEYS and XCSFKEY profiles can contain an ICSF segment. Fields of the ICSF segment specify rules for key use. In the SYMEXPORTABLE field of the ICSF segment, the BYLIST keyword specifies that the symmetric key or keys covered by the CSFKEYS or the XCSFKEY profile can be exported by keys identified using the SYMEXPORTKEYS or SYMEXPORTCERTS fields.

Using the SYMEXPORTCERTS field, you can supply a list of certificate labels in a trusted certificate repository (either a PKCS #11 token or a SAF key ring). As described in "Enabling PKA key management extensions" on page 58, you enable the PKA Key Management Extensions control by creating a CSF.PKAEXTNS.ENABLE profile in class XFACILIT. You can use the APPLDATA field in that profile to identify the type and name of the trusted certificate repository. If the APPLDATA field is not used to provide this information, the default certificate repository is a PKCS #11 token named CSF.TRUSTED.KEYRING. The format of the SYMEXPORTCERTS field depends on whether the trusted certificate repository is a PKCS #11 token or a SAF key ring.

- If the trusted certificate repository is a PKCS #11 token, the certificate labels are listed in the format '*cka-id/cert-label*', where:

  cka-id is the CKA_ID attribute of the certificate object. This portion of the specification is optional, and only necessary if multiple certificate objects have the same CKA_LABEL. If provided, RACF will convert this portion of the specification into uppercase before storing it in the profile.

  /cert-label
  is the CKA_LABEL attribute of the certificate object. Note that the forward slash character (/) is required even if the optional *cka-id* portion of the specification is omitted. If this portion of the specification contains blank characters, the entire specification must be enclosed in single quotes.

- If the trusted certificate repository is a SAF key ring, the certificate labels are listed in the format '*userID/cert-label*', where:

  userID is the owner of the certificate. This portion of the specification is optional, and only necessary if multiple certificates have the same label. If provided, RACF will convert this portion of the specification into uppercase before storing it in the profile.

/cert-label
: is the label of the digital certificate that was assigned when the certificate was created. Note that the forward slash character (/) is required even if the optional *userID* portion of the specification is omitted. If this portion of the specification contains blank characters, the entire specification must be enclosed in single quotes.

Regardless of whether you are using a PKCS #11 token or a SAF key ring, you can also use the asterisk character (*) in the SYMEXPORTCERTS field to match any certificate in the trusted certificate repository. Using the asterisk character in the SYMEXPORTCERTS field is the same as listing all the certificates in the trusted certificate repository.

If an application attempts to use the Symmetric Key Export (CSNDSYX or CSNFSYX) callable service to transfer a symmetric (AES or DES) key covered by the profile, ICSF will compare the RSA public key identified by the application with those bound to identities in certificates in the SYMEXPORTCERTS list. If any of the listed certificates contains the RSA public key, the operation is allowed to continue. If none of the listed certificates contain the public key, and the key is also not listed in the SYMEXPORTKEYS field (as described in "Identifying RSA public keys that can export the symmetric key" on page 55), the operation will fail and the service will return an error.

For example, say you want to allow export of a the symmetric key DES.BRADY.CASTLE only by the user and public key bound by a certificate in a SAF key ring. The SAF key ring was identified to ICSF when the PKA Key Management Extensions control was enabled (using the APPLDATA field of the CSF.PKAEXTNS.ENABLE profile). The label of the digital certificate in the SAF key ring is "Mister Ink", and the discrete profile covering the key has already been defined in the XCSFKEY class. The following RALTER command specifies that the only RSA public key that can export the symmetric key is the one bound to the identity in the "Mister Ink" certificate. The SETROPTS RACLIST command is used to refresh the profile in common storage.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYLIST) SYMEXPORTCERTS('/Mister Ink'))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

The preceding example assumes that no other certificates have the same label. If other certificates do have the same label, you would want to include the user ID of the certificate owner in the SYMEXPORTCERTS list specification. For example, if the user BKNIGHT is the certificate owner, you would enter:

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYLIST) SYMEXPORTCERTS('BKNIGHT/Mister Ink'))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

You can also use the asterisk character (*) in the SYMEXPORTCERT field to match any certificate in the certificate repository.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYLIST) SYMEXPORTCERTS(*))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

The ADDSYMEXPORTCERTS keyword of the ICSF segment enables you to add certificate labels to a SYMEXPORTCERTS list without having to re-create the entire list. For example, to add the certificate 'SERRIN/Mister Ink' to the list of certificate labels, you would enter:

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(ADDSYMEXPORTCERTS('SERRIN/Mister Ink'))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Similarly, you can delete certificate labels from a SYMEXPORTCERTS list using the DELSYMEXPORTCERTS keyword:

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(DELSYMEXPORTCERTS('BKNIGHT/Mister Ink'))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

You can also delete the entire SYMEXPORTCERTS field using the NOSYMEXPORTCERTS keyword.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(NOSYMEXPORTCERTS)
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Placing no additional restrictions on symmetric key export

If no keyword value is specified in the ICSF segment's SYMEXPORTABLE field, then, by default, no additional restrictions are placed on the export of symmetric keys covered by the profile. Provided no other access requirement or control prevents it, the symmetric key can be exported by any RSA public key. Although this is the default behavior, you can also explicitly specify it using the BYANY keyword. You might want to do this, for example, if you had previously specified the BYNONE or BYLIST keyword in the SYMEXPORTABLE field, and now want to return to the default behavior.

For example, to specify that there are no restrictions on the export of the symmetric key covered by the profile DES.BRADY.CASTLE in the XCSFKEY class, and that any RSA key can be used in the export operation (provided the user has access permission to the key), you could enter the following RALTER command. The SETROPTS RACLIST command is used to refresh the profile in common storage.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYANY))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

You can also return to the default behavior by deleting the entire SYMEXPORTABLE field using the NOSYMEXPORTABLE keyword.

```
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(NOSYMEXPORTABLE)
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Enabling PKA key management extensions: The rules for cryptographic key usage defined in the ICSF segment of CSFKEYS and XCSFKEY profiles (described in “Restricting asymmetric keys from being used in secure import and export operations” on page 51, “Restricting asymmetric keys from being used in handshake operations” on page 52, and “Placing restrictions on exporting symmetric keys” on page 53) will not be in effect unless PKA Key Management Extensions are enabled. PKA Key Management Extensions cannot be enabled unless Key Store Policy is active for both the CKDS and PKDS.

Enabling any one of the following controls will activate Key Store Policy for a CKDS:

- CSF.CKDS.TOKEN.CHECK.LABEL.WARN
- CSF.CKDS.TOKEN.CHECK.LABEL.FAIL
- CSF.CKDS.TOKEN.NODUPLICATES

Enabling any one of the following controls will activate Key Store Policy for a PKDS:

- CSF.PKDS.TOKEN.CHECK.LABEL.WARN
- CSF.PKDS.TOKEN.CHECK.LABEL.FAIL
- CSF.PKDS.TOKEN.NODUPLICATES

The following table shows the controls for enabling PKA Key Management Extensions in either warning or fail mode. To enable one of the controls, create the appropriate profile in the XFACILIT class.

Table 33. Key Store Policy controls: The PKA Key Management Extensions controls

The existence of this resource profile in the XFACILIT class:	**Does this:**
CSF.PKAEXTNS.ENABLE.WARNONLY	Enables PKA Key Management Extensions in warning mode. The ICSF segment of CSFKEYS or XCSFKEY profiles will be checked to: • determine if a symmetric key can be exported, and, if so, which asymmetric keys can be used in the operation to re-encrypt the symmetric key. • determine if an asymmetric key can be used in secure export and import operations, or in handshake operations. However, because this is warning mode, ICSF will allow the operation to continue even if the ICSF segment indicates that the operation is not allowed.
CSF.PKAEXTNS.ENABLE	Enables PKA Key Management Extensions in fail mode. The ICSF segment of CSFKEYS or XCSFKEY profiles will be checked to: • determine if a symmetric key can be exported, and, if so, which asymmetric keys can be used in the operation to re-encrypt the symmetric key. • determine if an asymmetric key can be used in secure export and import operations, or in handshake operations. If the ICSF segment indicates that the operation is not allowed, the service returns with an error.

For example, you've already used the ICSF segment of profiles in the CSFKEYS or XCSFKEY class to define various restrictions on how keys covered by the profiles can be used. You're not certain that all applications at your installation are using the keys according to the new restrictions, and do not want to disrupt current work patterns at your installation. For this reason, you decide to allow a warning period during which you can identify noncompliant applications without causing application failure. To do this, you would:

1. Enable PKA Key Management Extensions in warning mode:
   ```
   RDEFINE XFACILIT CSF.PKAEXTNS.ENABLE.WARNONLY
   SETROPTS RACLIST(XFACILIT) REFRESH
   ```
2. Because you have enabled PKA Key Management Extensions in warning mode, ICSF will allow applications to use keys in ways that violate ICSF segment specifications. However, ICSF will generate SMF type 82 subtype 27 records for any violation. Using the information in these records, you can modify your installation's applications as needed.
3. When you are ready to move to a stricter implementation of the policy, you enable the PKA Key Management Extensions control for fail mode, and disable the one for warning mode.
   ```
   RDEFINE XFACILIT CSF.PKAEXTNS.ENABLE
   RDELETE XFACILIT CSF.PKAEXTNS.ENABLE.WARNONLY
   SETROPTS RACLIST(XFACILIT) REFRESH
   ```

   If you accidentally enable PKA Key Management Extensions in both warning and fail mode, the control for fail mode will take precedence.

As described in “Identifying key certificates for symmetric key export” on page 56, you can use the ICSF segment's SYMEXPORTCERTS field to provide a list of certificate labels in a trusted certificate repository (either a PKCS #11 token or a SAF key ring). This enables you to specify that symmetric keys covered by a CSFKEYS or XCSFKEY profile can be exported only by RSA public keys that are bound to identities in the listed certificates. If using the SYMEXPORTCERTS field to provide a list of certificate labels in a trusted certificate repository, you will need to identify that trusted certificate repository to ICSF. You do this using the APPLDATA field of the CSF.PKAEXTNS.ENABLE profile. If the trusted key repository is a PKCS #11 token, it should be identified in the APPLDATA field in the format *TOKEN*/*PKCS-token-name*. If the trusted key repository is a SAF key ring, it should be identified in the APPLDATA field in the format *userID*/*key-ring-name*. For example, if the trusted key repository was a SAF key ring named TRUSTED.KEY.EXPORTERS created by BOBADMIN, you would enter:

```
RDEFINE XFACILIT CSF.PKAEXTNS.ENABLE APPLDATA(BOBADMIN/TRUSTD.KEY.EXPORTERS)
SETROPTS RACLIST(XFACILIT) REFRESH
```

If an APPLDATA field is not provided on the CSF.PKAEXTNS.ENABLE, the default certificate repository is a PKCS #11 token named CSF.TRUSTED.KEYRING.

PKA key management extensions example: The following example provides additional illustration of the ICSF segment fields and keywords that you can use to place restrictions on how cryptographic keys can be used.

A DES key has been created for encrypting transactions between a Company and its Business Partner. The Business Partner's public key has previously been added to the PKDS for the purpose of exporting the DES key. The Company's security administrator wants to be sure that only the Business Partner's public key can be used to export the DES key that the Company and its Business Partner are sharing. There is already a profile covering the label of the RSA public key in the PKDS, but no profile covering the label of the new DES key. The security administrator needs to alter the profile for the RSA public key label, and define a new profile for the DES key label. The security administrator has also enabled the Symmetric Key Label Export Control to increase the level of authority needed to export symmetric keys, and so the profile covering the DES key is defined in the XCSFKEY class.

```
RALTER CSFKEYS RSA.BRADY.CASTLE ICSF(ASYMUSAGE(SECUREEXPORT NOHANDSHAKE))
RDEFINE XCSFKEY DES.BRADY.CASTLE ICSF(SYMEXPORTABLE(BYLIST) SYMEXPORTKEYS(RSA.BRADY.CASTLE)) UACC(NONE)
PERMIT DES.BRADY.CASTLE CL(XCSFKEY) ID(SAMPRTNR) UPDATE
SETROPTS RACLIST(CSFKEYS) REFRESH
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Key Store Policy is active for both the CKDS and PKDS, so the security administrator only needs to enable the PKA Key Management Extensions control, and refresh the XFACILIT class in storage.

```
RDEFINE XFACILIT CSF.PKAEXTNS.ENABLE
SETROPTS RACLIST(XFACILIT) REFRESH
```

Later, the security administrator wants further restrictions on exporting the DES key that the Company and its Business Partner are sharing. The security administrator wants to bind an existing RSA public key to an identity, and allow export of the DES key only by the user and public key bound by a particular certificate. The security administrator creates the certificate for the RSA key, creates a SAF key ring, and adds the certificate to the key ring.

```
RACDCERT ID(BOBADMIN) GENCERT                                           +
         SUBJECTSDN(CN('Mister Ink Inc')O('Business Partner')C('uk'))  +
         WITHLABEL('Mister Ink')SIGNWITH(CERTAUTH LABEL(LocalCertauth')) +
         KEYUSAGE(DOCSIGN)                                              +
         NOTAFTER(DATE(2020-12-31))                                     +
         FROMICSF(RSA.BRADY.CASTLE)                                     +
RACDCERT ID(BOBADMIN) ADDRING(TRUSTD.KEY.EXPORTERS)
RACDCERT ID(BOBADMIN) CONNECT(LABEL('Mister Ink' RING(TRUSTD.KEY.EXPORTERS)  +
        USAGE(PERSONAL))
RALTER XCSFKEY DES.BRADY.CASTLE ICSF(NOSYMEXPORTKEYS                    +
        SYMEXPORTCERTS('/Mister Ink'))
SETROPTS RACLIST(XCSFKEY) REFRESH
```

Because the security administrator knows that only one certificate with the label "Mister Ink" will be present in the key ring, he does not specify the user ID portion of the string in the SYMEXPORTCERTS list. Note, however, that the security administrator still needs to include the forward slash (/) delimiter even though a user ID was not provided. Also note that the NOSYMEXPORTKEYS keyword is used to remove the SYMEXPORTKEYS list that had been previously defined.

The security administrator modifies the CSF.PKAEXTNS.ENABLE profile in the XFACILIT class to identify the SAF key ring as the certificate repository.

```
RDEFINE XFACILIT CSF.PKAEXTNS.ENABLE APPLDATA(TRUSTD.KEY.EXPORTERS)
SETROPTS RACLIST(XFACILIT) REFRESH
```

For more information on the ICSF fields and keywords, refer to “Restricting asymmetric keys from being used in secure import and export operations” on page 51, “Restricting asymmetric keys from being used in handshake operations” on page 52, and “Placing restrictions on exporting symmetric keys” on page 53.

Enabling use of archived KDS records: Records in the CKDS, PKDS, and TKDS can be marked as archived. A service request to use the key material of an archived record will fail and an SMF 82 audit record is then logged. The administrator, in an effort to remove unused records from the key data sets, may mark records that appear to have not been used in a long time as archived.

To prevent an application failure due to a rarely used label being marked as archived, the administrator can enable the Key Archive Use control. All service requests using archived records succeed and a SMF 82 record is logged. The administrator can check the SMF records for archived records that have been used. The administrator can also cause a joblog message to be issued by enabling the KEYARCHMSG control in the options data set.

For example, to enable the key archive use control for all key data sets, enter the following commands:

```
RDEFINE XFACILIT CSF.KDS.KEY.ARCHIVE.USE
SETROPTS RACLIST(XFACILIT) REFRESH
```

Distributing CCA keys

With ICSF, you can develop key distribution systems as defined in any of these:
- The IBM Common Cryptographic Architecture.
- ANSI TR-31 Key block.
- The Public Key Cryptographic Standard (PKCS).

These key distribution systems are explained in these topics:
- “Common Cryptographic Architecture Key Distribution”

Common Cryptographic Architecture Key Distribution

ICSF provides protection for keys when the keys are sent outside your system. You must generate complementary keys for key distribution. A complementary pair of keys has these characteristics:
- The keys have the same clear key value.
- The key types are different but complementary.

- Each key usually exists on a different system.

Some of the complementary keys are these types:
- Importer key-encrypting key and exporter key-encrypting key (transport keys).
- PIN generation key and PIN verification key.
- Input PIN-encrypting key and output PIN-encrypting key.
- MAC generation key and MAC verification key.
- Data-encrypting key and cipher text translation key.
- Input data-encrypting key and output data-encrypting key.
- Input key translate and output key translate keys

When protected data is sent between intermediate systems, these keys exist as complementary keys:
- Data-encrypting key and output cipher text translation key.
- Input cipher text translation key and output cipher text translation key.

For more information, see "Protection of data" on page 29.

The same data-encrypting key can also exist on two different systems so that both systems can encipher and decipher the data.

You can use ICSF to protect keys that are distributed across networks. You distribute keys across a network for some of these reasons:
- When you send encrypted data to another system, you send the data-encrypting key with the data or before it.
- When you share complementary keys with another system.

Transport keys protect keys being sent to another system. When a key leaves your system, an exporter key-encrypting key encrypts the key. When another system receives the key, the key is still encrypted under the same key-encrypting key, but the key-encrypting key is now considered an importer key-encrypting key. The exporter key-encrypting key at the sending system and the importer key-encrypting key at the receiving system must have the same clear value. For two systems to exchange keys, they must establish pairs of transport keys.

In Figure 6 on page 63 System A wants to send an output PIN-encrypting key to System B.

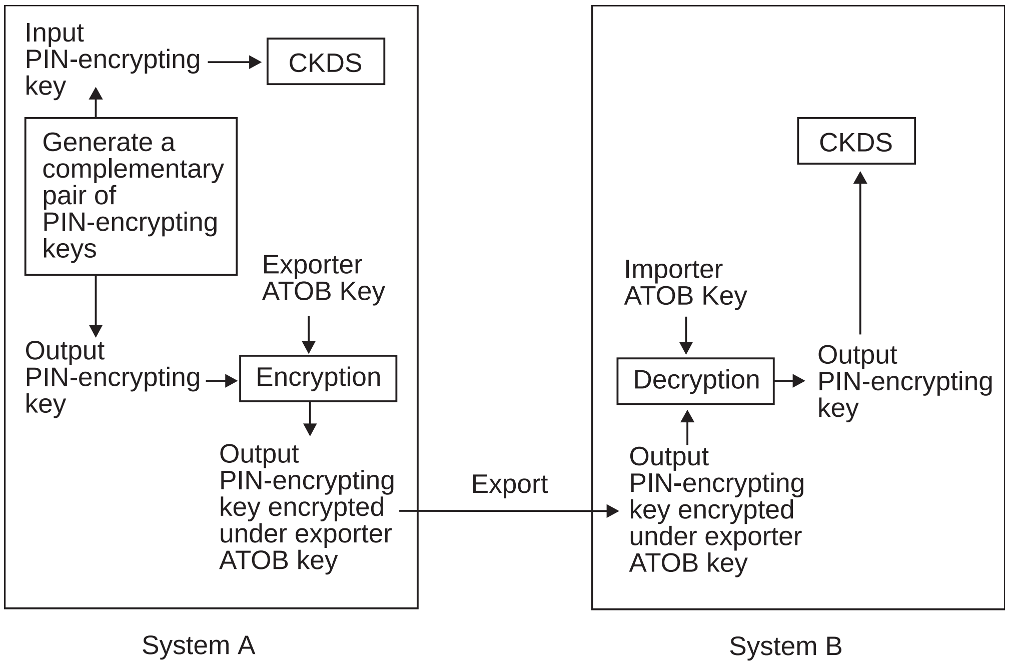

Input PIN-encrypting key
CKDS
Generate a complementary pair of PIN-encrypting keys
Exporter ATOB Key
Output PIN-encrypting key
Encryption
Output PIN-encrypting key encrypted under exporter ATOB key
Export
CKDS
Importer ATOB Key
Decryption
Output PIN-encrypting key
Output PIN-encrypting key encrypted under exporter ATOB key
System A
System B

Figure 6. Key Sent from System A to System B

To send the key, System A and System B must establish a pair of transport keys between them. System A has an exporter key-encrypting key called Exporter ATOB, which has the same key value as the importer key-encrypting key called Importer ATOB at System B. This pair of transport keys is unidirectional, because they are used only for distributing keys from System A to System B.

When System A generates the input PIN-encrypting key, the system also creates a complementary output PIN-encrypting key. System A enciphers the input PIN-encrypting key under System A's master key and stores the input PIN-encrypting key in the CKDS. It encrypts the complementary output PIN-encrypting key under the Exporter ATOB key so it can send the output PIN-encrypting key to System B. System B decrypts the output PIN-encrypting key using the Importer ATOB key, and encrypts the output PIN-encrypting key under System B's master key.

For the systems to send keys in both directions, they must establish two pairs of transport keys at each site, as in Figure 7.

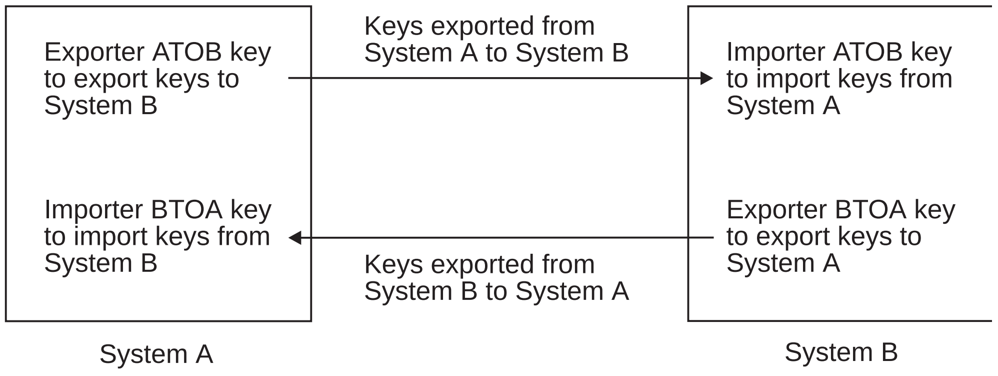

Exporter ATOB key to export keys to System B
Keys exported from System A to System B
Importer ATOB key to import keys from System A
Importer BTOA key to import keys from System B
Keys exported from System B to System A
Exporter BTOA key to export keys to System A
System A
System B

Figure 7. Keys Sent between System A and System B

To send keys from System A to System B, use the key generator utility program (KGUP) to establish an importer and exporter complementary key pair. You establish an exporter key, Exporter ATOB key, on System A and establish the complementary importer key, Importer ATOB key, on System B. Then when System A sends a key to System B, System A sends the key in exportable form encrypted under Exporter ATOB key. When System B receives the key, System B considers the key in importable form encrypted under Importer ATOB key.

To send keys from System B to System A, use KGUP to establish an importer and exporter complementary key pair. You establish an exporter key, Exporter BTOA key, on System B and the complementary importer key, Importer BTOA key, on System A. When System B sends a key to System A, System B sends the key in exportable form encrypted under Exporter BTOA key. When System A receives the key, System A considers the key in importable form encrypted under Importer BTOA key.

KGUP can create a pair of complementary keys, one key in operational form, and its complement in exportable form. You can also use KGUP to receive keys that are in importable form. When you want KGUP to create a key value in exportable form or import a key value in importable form, you specify the transport key that encrypts the key value. For more information about using KGUP for key distribution, see Chapter 11, “Managing Cryptographic Keys Using the Key Generator Utility Program,” on page 185.

You can also use one of two callable services to reencipher a key from operational form into exportable form. Both the key export callable service and the data key export callable service reencipher a key from encryption under the master key to encryption under an exporter key-encrypting key.

You can call the key import callable service to convert a key from importable form to operational form. The key import callable service reenciphers a key from encryption under an importer key-encrypting key to encryption under the system's master key.

With interlinked computer networks, sensitive data passes through multiple nodes before reaching its final destination. The originator and the receiver do not share a common key. Data-translation keys are shared between the originator and an intermediate system, between two intermediate systems, and between an intermediate system and the receiver system. As the data is passed along between these systems, they must reencipher it under the different data-translation keys without it ever appearing in the clear. Each system can call the ciphertext translate callable service to do this function. For a description of sending data between intermediate systems, see “Protection of data” on page 29.

ANSI TR-31 key block

ICSF provides support for importing and exporting DES keys using the American National Standards Institute (ANSI) TR-31 key block. The TR-31 key block supports the interchange of keys in a secure manner with key attributes included in the exchanged data. The TR-31 key block format has a set of defined key attributes that are securely bound to the key so that they can be transported together between any two systems that both understand the TR-31 format. ICSF enables applications to convert a CCA token to a TR-31 key block for export to another party and to convert an imported TR-31 key block to a CCA token. This enables you to securely exchange keys and their attributes with non-CCA systems.

Although there is often a one-to-one correspondence between TR-31 key attributes and the attributes defined by CCA, there are also cases where the correspondence is many-to-one or one-to-many. Because there is not always a one-to-one mapping between the key attributes defined by TR-31 and those defined by CCA, the TR-31 Export callable service and the TR-31 Import callable service provide rule array keywords enable an application to specify the attributes to attach to the exported or imported key.

Public Key Cryptographic Standard Key Distribution

ICSF provides support for the Public Key Cryptographic Standard (PKCS). PKCS is a set of standards for public-key cryptography developed by RSA Data Security, Inc. An example of using RSA public-key cryptography to distribute DES and AES data-encrypting keys is shown in “Using RSA public keys to protect keys sent between systems” on page 28.

Managing PKCS #11 cryptographic keys

PKCS #11 Overview

PKCS #11 is a standard set of programming interfaces for cryptographic functions developed by RSA Laboratories of RSA Security Inc. A subset of these functions is supported by ICSF. ICSF stores the PKCS #11 tokens and token objects in the TKDS. In the context of PKCS #11, a token is a representation of a cryptographic device, such as a smart card reader. You can store, update, and use public key objects, private key objects, secret key objects, certificate objects, data objects, and domain parameter objects in the TKDS through the use of PKCS #11 'C' API or ICSF's native callable services.

For more information on using the TKDS services, see *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* and *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Enterprise PKCS #11 master key

PKCS #11 key objects may be in either clear or secure (encrypted) format depending on your business needs. Secure keys require an active Enterprise PKCS #11 (EP11) Cryptographic Coprocessor. The coprocessor's master key (the P11 master key) is used to protect the sensitive key material. Clear keys are not protected by a master key.

The first time you start ICSF on your system, you may enter master keys and initialize the token data set (TKDS). You can then generate and enter the keys you use to perform cryptographic functions. The master keys you enter protect the secure keys stored in the TKDS.

The TKE workstation must be used to enter P11 master keys on the EP11 cryptographic coprocessors. The TKE workstation is an optional hardware feature. The TKE workstation uses a variety of public key cryptographic techniques to ensure both the integrity and privacy of the logically secure master key transfer channel. You can use a single TKE workstation to set up master keys in all EP11 coprocessors within a server complex.

For more information on using the TKE workstation, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Note: Servers or processor models may have multiple EP11 cryptographic coprocessors. Additionally, the TKDS may be shared by multiple systems or LPARs. The master keys must be the same for all such coprocessors accessed by the system or systems sharing the TKDS.

Managing tokens and objects in the TKDS

Because PKCS #11 is a standard application programming interface, the expected way to manage tokens and objects in the TKDS is through the invocation of the PKCS #11 'C' API or ICSF's native callable services. However, ICSF does provide an ISPF panel utility to allow you to query and make some minor modifications to tokens and objects stored in the TKDS. This utility is known as the PKCS11 Token Browser.

The PKCS11 Token Browser utility allows you to:

- View the tokens in the TKDS.
- Create tokens.
- Delete tokens.
- View the objects of a token.
- Delete objects.
- Modify object attributes.

SAF authority to the PKCS #11 tokens is required to view and manage tokens and objects. See Chapter 16, "Using PKCS11 Token Browser Utility Panels," on page 325 for additional information.

PKCS #11 and FIPS 140-2

The National Institute of Standards and Technology (NIST), the US federal technology agency that works with industry to develop and apply technology, has published the Federal Information Processing Standard #140-2 (FIPS 140-2) *Security Requirements for Cryptographic Modules* that can be required by organizations who specify that cryptographic-based security systems are to be used to provide protection for sensitive or valuable data.

The z/OS PKCS #11 services are designed to meet FIPS 140-2 Level 1 criteria and can be configured to operate in compliance with FIPS 140-2 specifications. Applications that need to comply with the FIPS 140-2 standard can therefore use the z/OS PKCS #11 services in a way that allows only the cryptographic algorithms (including key sizes) approved by the standard and restricts access to the algorithms that are not approved.

For more information on using the PKCS #11 services, refer to the *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* and the *z/OS Cryptographic Services ICSF System Programmer's Guide*.

TKDS key protection

The TKDS may contain both clear and secure keys. Clear keys stored in the TKDS are not encrypted. Therefore, it is recommended that you SAF-protect data set access to the TKDS. (This is in addition to the SAF protection of the individual tokens via the CRYPTOZ class.) This will provide additional security for your installation.

Managing regional cryptographic keys

ICSF's regional cryptographic support is an extension to the standard PKCS #11 services. To manage regional cryptographic keys, you must be familiar with how to manage PKCS #11 keys in general. Read “Managing PKCS #11 cryptographic keys” on page 65 before continuing with this section.

Persistent regional cryptographic keys are stored as PKCS #11 key objects inside tokens in the TKDS. Therefore, the PKCS11 Token Browser ISPF panel utility may be used to manage regional cryptographic keys.

Regional cryptographic server master key

As with Enterprise PKCS #11 keys, regional cryptographic keys are secure keys, protected by a master key. This master key is called the regional cryptographic server master key (RCS-MK). This is a different master key from the P11-MK and is managed differently. Unlike the P11-MK, ICSF is minimally involved in the management of the RCS-MK. The RCS-MK is loaded on the server using a utility provided by the vendor of the device. For ICSF, there is no explicit initialization step. Initialization implicitly happens the first time that a regional cryptographic server is used by ICSF. The first RCS-MK that ICSF encounters becomes the expected RCS-MK going forward. Furthermore, each regional cryptographic server that ICSF is to connect to must have the same RCS-MK set. This requirement extends to a SYSPLEX environment as well. All regional cryptographic servers being used by any instance of ICSF sharing a given TKDS must have the same RCS-MK set.

Chapter 4. Setting up and maintaining cryptographic key data sets

The cryptographic key data sets store keys for use by ICSF callable services. The services use a label to identify the record to be used. This topic discusses the setting up and maintenance of the key data sets.

In addition, each key data set record has metadata when using the KDSR format of the key data set. The metadata can be used to determine if a record has been used to set up key material validity dates, archive and recall records, and store installation data for a record.

Setting up and maintaining the cryptographic key data set (CKDS)

The cryptographic key data set (CKDS) stores operational DES, AES, and HMAC keys of all types. It contains an entry for each key. An installation is not required to define a CKDS. However, when a CKDS is not defined, secure CCA symmetric key functions are unavailable and ICSF cannot be used to manage CCA symmetric key tokens.

There are three formats of the CKDS:

- A fixed-length record format that is supported by all releases of ICSF.
- A variable length record format that is supported by ICSF FMID HCR7780 and later releases.
- A common variable length record format that is supported by ICSF FMID HCR77A1 and later releases. This format is referred to as KDSR format.

It is recommended that you use the common record format (KDSR) for all your key data sets. KDSR format supports additional functions to manage cryptographic keys. For information on converting your existing CKDS to KDSR format, see 'Migrating to the KDSR format key data set' in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

If you have no coprocessor, you can initialize the CKDS for use with clear AES and DES data keys. This CKDS cannot be used on a system with cryptographic coprocessors.

Before you generate keys that you store in the CKDS, you must define a DES or AES master key to your system. You define a master key by entering its value and setting it so it is active on the system. When you enter the master key, you must make it active on the system by setting it when you initialize the CKDS. For information about entering and setting the master key and initializing CKDS, see Chapter 7, “Managing CCA Master Keys,” on page 103.

DES keys that are stored in the CKDS are encrypted under the appropriate variants of the DES master key, except for clear key value data-encrypting keys. AES keys that are stored in the CKDS are encrypted under the AES master key. HMAC keys are encrypted under the AES master key. Encrypted keys in the CKDS cannot be overwritten with a key encrypted under a different master key. (DES replaces DES, AES replaces AES, HMAC replaces HMAC). For clear keys, the same is true: DES can overwrite DES, AES can overwrite AES, and HMAC can overwrite HMAC.

After you define a master key, you generate keys and store them in the CKDS. You use KGUP to generate keys and change key values and other information for a key entry in the CKDS. For more information about running KGUP, see Chapter 11, “Managing Cryptographic Keys Using the Key Generator Utility Program,” on page 185. You can also program applications to use callable services to generate keys and change key information in the CKDS. For more information about how to use callable services to update key entries in the CKDS, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

You can load key parts for all operational keys on the coprocessors by using the TKE workstation. To load the accumulated key into the CKDS, you must use the ICSF Operational Key Load panel or KGUP. For more information, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

When you initialize ICSF, the system obtains space in storage for the CKDS. For more information about initializing space for the CKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Besides the in-storage CKDS, there is a copy of the CKDS on disk. Your installation can have many CKDS disk copies, backup copies, and different disk copies. For example, an installation can have a separate CKDS with different keys for each shift. When a certain shift is working, you can load the CKDS for that shift into storage. Then, only the keys in the CKDS loaded for that shift can be accessed for ICSF functions. However, only one disk copy is read into storage at a time.

You use KGUP to make changes to any disk copy of the CKDS. When you use KGUP to generate and maintain keys, or enter keys directly, you change only the disk copy of a CKDS. Therefore, you can change keys in the disk copy of the data set without disturbing ICSF functions that are using the keys in the in-storage copy of the data set. To make the changes to the disk copy of the CKDS active, you need to replace the in-storage CKDS by using the refresh utility. When you use the dynamic CKDS update callable services to change entries in the CKDS, you change both the in-storage copy of the CKDS and the disk copy. This allows for the immediate use of the new keys without an intervening refresh of the entire CKDS. Figure 8 on page 71 shows the ICSF callable services use keys in the in-storage copy of the CKDS.

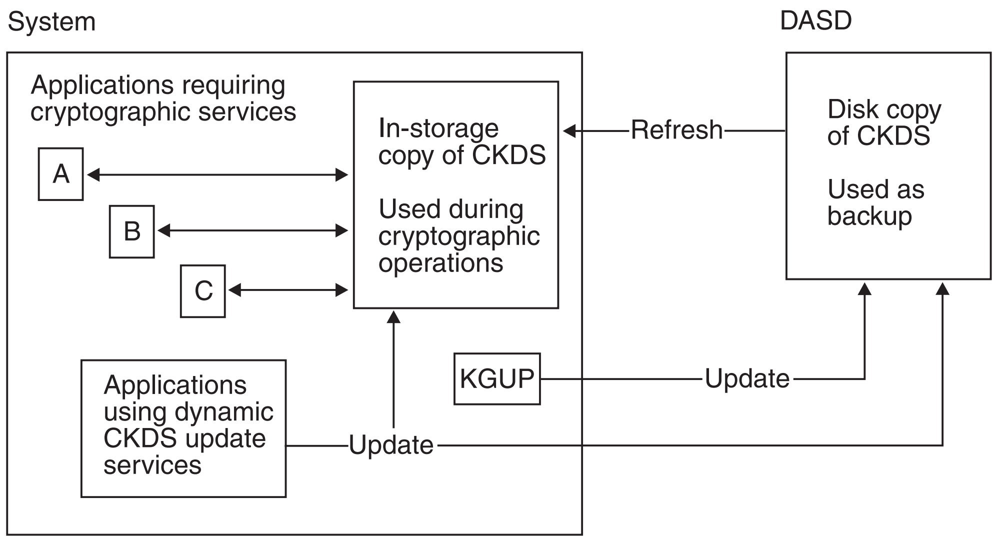

System
DASD
Applications requiring cryptographic services
A
B
C
In-storage copy of CKDS
Used during cryptographic operations
Refresh
Disk copy of CKDS
Used as backup
KGUP
Update
Applications using dynamic CKDS update services
Update

Figure 8. Updating the in-storage copy and the disk copy of the CKDS

You specify the name of the disk copy of the CKDS when you run KGUP. You can also read any disk copy of the CKDS into storage by specifying the name of the disk copy of the CKDS on a Refresh In-Storage CKDS panel. You can also run a utility program to read a disk copy of the CKDS into storage. However, the disk copy must be enciphered under the correct master key. All the copies of your disk copies of the CKDS should be enciphered under the same master key.

Your installation should periodically change the symmetric master keys: DES and AES. To change a master key, you enter a new master key value and make that value active. The keys in a CKDS must then be enciphered under the new master keys. Therefore, to make the new master keys active, the CKDS must be reenciphered from under the current master keys to under the new master keys.

There are two ways to change the symmetric master keys. The preferred way to perform a master key change is by using the Coordinated CKDS Change MK function. This function is described in “Performing a coordinated change master key” on page 147.

Optionally, the symmetric master keys can be changed on a single system. To perform a local symmetric change master key, first you reencipher the disk copy of the CKDS under the new master keys. Then, you activate the new master keys by using the change master key option. This option automatically replaces the old in-storage CKDS with the disk copy that is reenciphered under the new master keys. If you have multiple CKDS disk copies, reencipher all of them under the new master keys before changing the master key.

The local symmetric change master key change process can be accomplished by using either the options on the ICSF CKDS Master Key Management panels or by using the utility program, CSFEUTIL.

Note: When you perform any functions that affect the in-storage copy of the CKDS, you should consider temporarily disallowing the dynamic CKDS update services. Functions that affect the in-storage copy of the CKDS include changing the master key, reenciphering, or refreshing. For more information, see “Steps for disallowing dynamic CKDS updates during CKDS administration updates” on page 186.

If running in a sysplex, see Chapter 10, “Running in a Sysplex Environment,” on page 171.

Unsupported keys in the CKDS

The symmetric algorithms and callable services that ICSF supports have changed with the cryptographic coprocessors available on z systems. There are keys associated with these algorithms and services that may be stored in the CKDS. These keys cannot be used for any other purpose. These keys may be removed if desired.

These symmetric keys are unsupported:

- DATAXLAT key used with the previously supported CSNBCTT service.
- ANSI X9.17 keys used with the with previously supported CSNAKEX, CSNAKIM, CSNAKTR, and CSNATKN services.

To determine if your CKDS has any unsupported keys:

- The ICSF_UNSUPPORTED_CCA_KEYS health check will list the records of all unsupported key in the active CKDS. This health check runs every time ICSF is started. The check will list records regardless of the state of the record. Archived and inactive records will appear in the Health Check output.
- The CSFKDSL callable service can be used to get a count or a list of records with unsupported keys in the active CKDS.

For installations using the common record format (KDSR) of the CKDS:

- If you have key reference tracking enabled, you can tell if the label has been referenced by an application using the CSFKMDR callable service to read the last reference date metadata of the record.
- You can archive these records to see if they are being referenced by your applications before deleting the records. The CSFKMDW callable service is used to archive records.

The records can be deleted using the CSNBKRD service.

For the description of the services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Setting up and maintaining the public key data set (PKDS)

Public Key Algorithm (ECC and RSA) public and private keys and trusted blocks can be stored in the public key data set (PKDS), a VSAM data set. Applications can use the dynamic PKDS callable services to create, write, read, and delete PKDS records. An installation is not required to define a PKDS. However, when a PKDS is not defined, secure CCA asymmetric key functions are unavailable and ICSF cannot be used to manage CCA asymmetric key tokens.

There are two formats of the PKDS:

- A variable length record format supported by all releases of ICSF.
- A common variable length record format supported by ICSF FMID HCR77A1 and later releases. This format is referred to as KDSR format.

It is recommended that you use the common record format (KDSR) for all your key data sets. KDSR format supports additional functions to manage cryptographic keys. For information on converting your existing PKDS to KDSR format, see 'Migrating to the KDSR format key data set' in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

The PKDS may be initialized at ICSF setup. There are internal and external tokens in the PKDS. External tokens may be used irrespective of the asymmetric master keys. Internal tokens, however, can only be used if they are encrypted under the appropriate asymmetric master key.

Besides the in-storage PKDS, there is a copy of the PKDS on disk. Your installation may have many PKDS disk copies, backup copies, and different disk copies. For example, an installation may have a separate PKDS with different keys for each shift. When a certain shift is working, you can load the PKDS for that shift into storage. Then only the keys in the PKDS loaded for that shift can be accessed for ICSF functions. Only one disk copy is read into storage at a time.

Your installation should periodically change the asymmetric master keys. To change the master keys, you enter a new master key value and make that value active.

There are two ways to change the asymmetric master keys. The preferred way to change the master keys is by using the Coordinated PKDS Change MK function. For more information on this function. see “Performing a coordinated change master key” on page 147.

Optionally, the asymmetric master keys can be changed on a single system. To perform a local asymmetric master key change, the PKDS must be reenciphered under the new master keys. You can reencipher a PKDS under a new master key using the options on the ICSF PKDS Master Key Management panel or by using a utility program, CSFPUTIL. If you have multiple PKDS disk copies, reencipher all of them under the new master key after loading the new master key value.

You can program applications to use the PKDS callable services to create entries, change entries, and delete entries in the PKDS. For more information about how to use callable services to update key entries in the PKDS, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

PKDS key management panels support:
- Generating an RSA key pair PKDS record.
- Deleting an existing PKDS record.
- Exporting an existing public key to an X.509 certificate stored in an MVS physically sequential data set.
- Importing a public key from an X.509 certificate stored in an MVS physically sequential data set.

If running in a sysplex, see Chapter 10, “Running in a Sysplex Environment,” on page 171.

Unsupported keys in the PKDS

The asymmetric algorithms and callable services that ICSF supports have changed with the cryptographic coprocessors available on z Systems. There are keys that are associated with these algorithms and services that might be stored in the PKDS. These keys cannot be used for any other purpose. These keys can be removed if desired.

These symmetric keys are unsupported:

- DSS (DSA) public and private keys that are used with the CSNDDSG and CSNDDSV services.

To determine whether your PKDS has any unsupported keys:

- The ICSF_UNSUPPORTED_CCA_KEYS health check lists the records of all unsupported key in the active PKDS. This health check runs every time ICSF is started. The check lists records regardless of the state of the record. Archived and inactive records appear in the Health Check output.
- The CSFKDSL callable service can be used to get a count or a list of records with unsupported keys in the active PKDS.

For installations that use the common record format (KDSR) of the PKDS:

- If you have key reference tracking enabled, you can tell if the label has been referenced by an application by using the CSFKMDR callable service to read the last reference date metadata of the record.
- You can archive these records to see whether they are being referenced by your applications before deleting the records. The CSFKMDW callable service is used to archive records.

The records can be deleted by using the CSNDKRD service.

For the description of the services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Setting up and maintaining the token data set (TKDS)

Clear and secure PKCS #11 objects can be stored in the token data set (TKDS), a VSAM data set. Applications can use the PKCS #11 callable services to create, write, read, and delete PKCS #11 tokens and objects.

There are two formats of the TKDS:

- A variable length record format supported by all releases of ICSF.
- A common variable length record format supported by ICSF FMID HCR77A1 and later releases. This format is referred to as KDSR format.

It is recommended that you use the common record format (KDSR) for all your key data sets. KDSR format supports additional functions to manage cryptographic keys. For information on converting your existing TKDS to KDSR format, see 'Migrating to the KDSR format key data set' in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

By default, an empty TKDS is initialized for clear PKCS #11 usage the first time it is used. No manual initialization step is required. To use secure PKCS #11 services (either Enterprise PKCS #11 or regional cryptographic services), the TKDS must be initialized with the appropriate master key. The appropriate master key is the PKCS #11 master key (P11-MK) for Enterprise PKCS #11 services or the regional cryptographic server master key (RCS-MK) for regional cryptographic services. For the RCS-MK, initialization implicitly happens the first time a regional cryptographic server is used. As with clear keys, no manual initialization step is required.

To use Enterprise PKCS #11 services, the TKDS must be explicitly initialized by the ICSF TKDS Master Key Management panel. This initialization step may be performed against an empty TKDS or one that contains existing clear objects, regional cryptographic objects, or both. Initialization does not alter the existing key objects in any way. They are still usable for their supported key operations as before.

Your installation should periodically change the PKCS #11 master keys.

- To change the master key, first load a new P11-MK value using a TKE workstation.
- To change the RCS master key, first load a new RCS-MK value using your RCS vendor utility.

See Chapter 8, "Managing PKCS #11 master keys," on page 159 for more details. Then, make the new master key active by performing a Coordinated TKDS Change MK from the ICSF TKDS Master Key Management panel.

Note: All regional cryptographic servers used by instances of ICSF that share the TKDS must have the same RCS master key.

You can program applications to use the PKCS #11 callable services to create PKCS #11 tokens and objects and to perform PKCS #11 cryptographic operations. See *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* for details.

If running in a sysplex, see Chapter 10, "Running in a Sysplex Environment," on page 171.

Key data set metadata

Records in the key data sets have metadata which can be used to manage the life cycle of key material. The metadata can be used as search criteria to generate list of labels or objects. The metadata can be read and changed.

Note: Only the KDSR format of the key data sets support all the metadata described in the section. Your existing data sets can be converted to the KDSR format using ICSF panels or the Coordinated KDS Administration callable service in *z/OS Cryptographic Services ICSF Application Programmer's Guide*. If you do not need metadata support for all of your key data sets, you need only to convert those data sets which do need metadata support. All of this metadata applies to the records in the CKDS and PKDS and to the object records in the TKDS. While PKCS #11 token are stored in the TKDS, they do not have metadata.

Metadata

The following fields and blocks are metadata in the KDS record:

Record creation date
The date and time that the record was created in the KDS.

Record update date
The date and time of the last time that the key material or metadata of the record was changed.

Key material validity start date
The date that the key material becomes active.

Key material validity end date
The last date that the key material is active.

Last used reference date
The date that the key material was last referenced.

Record archive flag
: When enabled, the key material cannot be used when the record is referenced by an application.

Record archive date
: The date that the record archive flag was enabled by the KDS Metadata Write service.

Record recall date
: The date that the record archive flag was disabled by the KDS Metadata Write service.

Record prohibit archive flag
: When enabled, the record cannot be archived.

User data
: The data stored in the user data field in the old formats of the CKDS (CKDUDATA), PKDS (PKDUDATA), or TKDS (TKDUDATA).

| **Key fingerprint**
| : A set of identifiers for the key. Different key values are likely to have different identifiers, but that is not guaranteed. See *z/OS Cryptographic Services ICSF Application Programmer's Guide* for more information about the format of this metadata.

IBM and installation variable-length metadata blocks
: Blocks of metadata.

Record creation and update dates

Record creation and update dates can be used as search criteria when generating a list of records using the Key Data Set List service. These dates can be read by the Key Data Set Metadata Read service. These dates cannot be modified by the Key Data Set Metadata Write service.

Key material validity start and end dates

These dates can be used as search criteria when generating a list of records using the Key Data Set List service. These dates can be read by the Key Data Set Metadata Read service. These dates can be added, changed, and deleted using the Key Data Set Metadata Write service.

Last used referenced date

This date can be used as search criteria when generating a list of records using the Key Data Set List service. This date can be read by the Key Data Set Metadata Read service. This date can be added, changed, and deleted using the Key Data Set Metadata Write service.

Record archive and recall dates

These dates can be used as search criteria when generating a list of records using the Key Data Set List service. These dates can be read by the Key Data Set Metadata Read service. These dates are changed when the record archive flag is enabled or disabled using the Key Data Set Metadata Write service.

Record archive and record prohibit archive flags

The record prohibit archive flag can be used as search criteria when generating a list of records using the Key Data Set List service. These flags can be read by the Key Data Set Metadata Read service. These flags can be enabled and disabled using the Key Data Set Metadata Write service.

Key fingerprint

The key fingerprint cannot be used as search criteria with the Key Data Set List service. The value can be read by the Key Data Set Metadata Read service. The value cannot be updated using the Key Data Set Metadata Write service.

Variable-length metadata blocks

The metadata tags can be used as search criteria when generating a list of records using the Key Data Set List service. The metadata blocks can be read by the Key Data Set Metadata Read service. The metadata blocks can be added, changed, and deleted using the Key Data Set Metadata Write service.

Archiving and recalling a record in a key data set

Key administrators can mark a record as archived. ICSF generates an audit record when the record is archived and whenever an archived record is referenced by an application. The administrator can specify whether the request to reference an archived record will succeed or fail. A key is referenced when it is used to perform a cryptographic operation or read, such that the retrieved token may have been used in a cryptographic operation.

Note: Only the KDSR format of the key data sets support archiving records. Your existing data sets can be converted to the KDSR format using the Coordinated KDS Administration callable service in *z/OS Cryptographic Services ICSF Application Programmer's Guide*. If you do not need metadata support for all of your key data sets, you can convert only those data sets which need metadata support.

To archive a record, the record archive flag must be enabled using the Key Data Set Metadata Write service. A SMF type 82 record is generated. The record remains in the key data set. The record can be deleted by the Key Record Delete services. The key material and metadata are deleted when the record is deleted.

To recall an archived record, the record archive flag must be disabled using the Key Data Set Metadata Write service. A SMF type 82 record is generated.

If an application attempts to use an archived record, a SMF type 82 record is generated. The XFACILIT resource CSF.KDS.KEY.ARCHIVE.USE control (see "Key Store Policy" on page 39) determines whether the service request succeeds or fails. If the key archive use control is enabled, the request is allowed to succeed and the return code reflects the processing of the service. If the key archive use control is disabled, the request fails with a return code of 8 and a reason code indicating a record was archived.

In addition to the key archive use control, the key archive message control (KEYARCHMSG keyword in the options data set) causes a joblog message to be issued the first time an archived record is successfully used by an application.

Variable-length metadata blocks

Metadata can be stored in the key data set in the KDSR format. A 2-byte tag is used to identify a block of metadata. IBM reserves the tags where the high order bit is off. Tags reserved for installation metadata block will have the high order bit on.

All metadata block tags can be used as search criteria for the Key Data Set List service. All metadata block tags can be read using the Key Data Set Metadata Read service.

Table 34. Metadata tag usage

Tag usage	Value or range	Notes
The data stored in the user data field in the old formats of the CKDS (CKDUDATA), PKDS (PKDUDATA), or TKDS (TKDUDATA).	X'0001'	This metadata block can be added, deleted, and changed using the Key Data Set Metadata Write service.
The service or utility that referenced the record the last time the last used reference date was updated.	X'0002'	This metadata block is read only.
The date the record was archived.	X'0003'	This metadata block is read only.
The date the record was recalled.	X'0004'	This metadata block is read only.
The key fingerprint	X'0005'	This metadata block is read only.
Retained RSA key information	X'0006'	This metadata block is read only.
Reserved for IBM use.	X'0007' - X'7FFF'	These metadata blocks are read only.
Available for installation use.	X'8000' - X'FFFF'	Metadata blocks can be added, deleted, and changed using the Key Data Set Metadata Write service.

IBM metadata blocks

IBM has metadata blocks for:

- The date that the record was archived by the KDS Metadata Write service.
- The date that the record was recalled by the KDS Metadata Write service.
- The callable service or utility that was invoked the last time that the key material was used in a cryptographic operation. This field is updated when the last referenced date is updated.
- Data from the installation user data field in the old formats of the CKDS (CKDUDATA), PKDS (PKDUDATA), or TKDS (TKDUDATA).
- The key fingerprint associated with the key.

Installation metadata blocks

The maximum amount of storage available to installation metadata is 500 bytes. This includes the tag and length fields of the metadata blocks. The data stored in the block can be in any format. The data will not be checked by ICSF when added to a record.

Key material validity dates

Administrators can set start and end validity dates for a key data set record using the KDS Metadata Write service. The end date cannot be set to a date in the past.

If the key validity dates are set for a record, any service attempting to reference the key material checks that the current date and time (coordinated universal time (UTC)) falls within the validity dates. The record becomes active at 00:00 UTC on the start date and becomes inactive at 00:00 UTC on the day after the end date. The system clock is used for this test. If the system clock is set to local time, the time will be 00:00 local time. A key is referenced when it is used to perform a cryptographic operation or read, such that the retrieved token may have been used in a cryptographic operation.

If an application attempts to use an inactive record, a SMF type 82 record is generated and the service request fails.

The key material validity dates are checked before the record archived flag.

Sharing KDS with older releases of ICSF and sysplex implications

The KDSR format of the KDS was introduced in ICSF FMID HCR77A1. There is no support in ICSF FMID HCR77A1 to enforce key material validity dates or archived records. If you plan to share your key data sets with ICSF FMID HCR77A1 and plan to use the validity dates and archive records, understand that when your application executes on systems with ICSF FMID HCR77A1, the inactive and archived records are used as if they were not archived or have valid validity dates.

Adding cryptographic coprocessors after initialization

You may need to initialize coprocessors after system initialization.

Note: If you used the Pass Phrase Initialization (PPINIT) utility to initialize your system, use the utility to added coprocessors. See "Steps for adding a CCA coprocessor after first time Pass Phrase Initialization" on page 100.

This is the procedure:

1. Load the new master key register with the current master key values using the ICSF Master Key Entry panel (see "Steps for entering the first master key part" on page 117) or the TKE workstation.
2. When the new master key register is loaded, SET the master key. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel: "ICSF Primary Menu panel" on page 387

   The "CSFMKM10 — Key Data Set Management panel" on page 390 appears.
3. Choose option 4, SET MK.

   The master keys will be set if the verification pattern of the new master key register matches the MKVP in the key data sets. All new master key registers that match will be set.

Chapter 5. Controlling who can use cryptographic keys and services

System authorization facility (SAF) controls

You can use a System Authorization Facility such as z/OS Security Server RACF to control which applications can use specific keys and services. This can help you ensure that keys and services are used only by authorized users and jobs. You can also use SAF to audit the use of keys, services, and utilities. To use SAF to control access to keys, services, and utilities, you create and maintain general resource profiles in the CSFKEYS, XCSFKEY, CSFSERV, and CRYPTOZ classes.

Notes:

- The CSFKEYS class grants access to the key if there is no profile covering the key label.
- The CSFSERV class grants access to the service if there is no covering profile, but with the following exceptions:
  - Key Data Set Update (CSFKDU and CSFKDU6)
  - Key Data Set Record Retrieve (CSFRRT and CSFRRT6)
- The XCSFKEY class does not grant access to the resource if there is no profile.
- The CRYPTOZ class does not grant access to the resource if there is no profile.
- The CSFSERV class controls access to CCA and PKCS #11 services and ICSF TSO panel utilities.
- The CSFKEYS class controls access to CCA cryptographic keys. You create profiles in this class (based on the label by which the key is defined in the CKDS or PKDS) to set access authority for the keys.
- The XCSFKEY class controls authorization checks when the symmetric key export services are called. See "Increasing the level of authority required to export symmetric keys" on page 48 for additional information.
- The CRYPTOZ class controls access to and defines policy for cryptographic information within PKCS #11 tokens. PKCS #11 tokens are used exclusively by ICSF's PKCS #11 callable services. They are abstract containers that hold keys, certificates, and other related cryptographic information. PKCS #11 tokens are usually explicitly created and assigned to specific applications. The profiles in the CRYPTOZ class determine this assignment and indicate what operations are permitted for a specific PKCS #11 token.

If you are not the security administrator, you may need to ask assistance from that person. To use the auditing capabilities of SAF, you may need to ask for reports from a SAF auditor. Your installation's security plan should show who is responsible for maintaining these SAF profiles and auditing their use.

Cryptographic coprocessor access controls for services and utilities

In addition to the CSFSERV class, CCA and PKCS #11 services and utilities are controlled by access control points in the domain role of all cryptographic coprocessors. Most access control points are enable by default. Some access control points are disabled by default for all users and require a TKE workstation to enable.

The access control points are listed in Appendix E, "CCA access control points and ICSF utilities," on page 407 and Appendix G of *z/OS Cryptographic Services ICSF Application Programmer's Guide*. The PKCS #11 access control points are listed in Chapter 2 of *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

All access control points for ISPF, UDX, and callable services on the coprocessor can be enabled or disabled using the TKE workstation. A TKE workstation is required if you are using the Enterprise PKCS #11 coprocessor and PKCS #11 access control points may be enabled or disabled.

When a new release of licensed internal code (LIC) is installed on a coprocessors and there are new access control points:

- If you do not have a TKE workstation, the new CCA access control points will have the default setting from the LIC.
- If you have a TKE workstation and you have not changed the settings of any CCA access control points, the new CCA access control points will have the default setting from the LIC.
- If you have a TKE workstation and you have changed the settings of any CCA access control points, the new CCA access control points will be disabled.
- New PKCS #11 access control points will be disabled.

New access control points must be enabled before the new services are available. UDX support is dependent on access control points. If your installation wants to use UDX callable services, the corresponding access control point must be enabled.

For more information, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Steps for SAF-protecting ICSF services and CCA keys

This procedure describes one approach for SAF-protecting services and CCA keys:

1. Decide whether you will protect keys, services, or both. You can select which keys and services to protect. The CSFKEYS, CSFSERV, and XCSFKEY classes grant access to the key or service if there is no profile. You can create a global generic profile to restrict access.
2. You may want to organize the users who need access to ICSF keys and services into groups. To do this, obtain a list of the user IDs of users who need to use ICSF keys and services. If batch jobs or started tasks need to use ICSF, obtain the user IDs under which they will run.

   Group any of the user IDs together if they require access to the same keys and services. For example, you might want to set up groups as follows:

   - Users who work with MAC-related callable services.
   - Users who work with a particular MAC or a particular PIN.
   - Users who call applications to dynamically update the CKDS or PKDS.
   - Users who work with digital signing callable services.
   - Users who work with PKCS #11 applications.

   Usually, all users of ICSF should have access to keys and services by virtue of their membership in one of these SAF groups, rather than specific users. This is because SAF maintains the access lists in in-storage profiles. When the in-storage profiles are created or changed, the in-storage profiles must be refreshed. (Merely changing them in the SAF database is not sufficient. This is analogous to the in-storage CKDS maintained by ICSF.) To refresh the in-storage SAF profiles, the security administrator must use the SETROPTS command:

```
SETROPTS RACLIST(CSFKEYS) REFRESH
SETROPTS RACLIST(CSFSERV) REFRESH
```

If you place *SAF groups* in the access lists of the SAF profiles, you can change a user's access to the protected services and keys by adding or removing the user from the groups. Ask your security administrator to create the SAF groups.

You should also ask your security administrator to connect you to these groups with CONNECT group authority. This permits you to connect and remove users from the groups.

For example, the security administrator could issue these commands:

```
ADDGROUP groupid
CONNECT  your-userid  GROUP(groupid)  AUTHORITY(CONNECT)
```

With CONNECT group authority, you are able to connect other users to the groups:

```
CONNECT  other-userid  GROUP(groupid)
```

With CONNECT group authority, you are also able to remove users from the groups:

```
REMOVE  other-userid  GROUP(groupid)
```

3. Ask your security administrator for the authority to create and maintain profiles in the CSFKEYS and CSFSERV general resource classes. Usually, this is done by assigning a user the CLAUTH (class authority) attribute in the specified classes. For example, the security administrator can issue this command:

   ```
   ALTUSER  your-userid  CLAUTH(CSFKEYS CSFSERV)
   ```

4. If you want to use generic profiles that contain characters such as * and %, ask your security administrator to activate generic profile checking in the CSFKEYS and CSFSERV classes:

   ```
   SETROPTS  GENERIC(CSFKEYS CSFSERV)
   ```

   Note: Using generic profiles has several advantages. Using generic profiles, you can reduce the number of profiles that you need to maintain. You can also create a "top" generic profile that can be used to protect all keys and services that are not protected by a more specific profile.

5. Define profiles in the CSFKEYS and CSFSERV classes. For further instructions, see "Setting up profiles in the CSFKEYS general resource class" on page 93 and "Setting up profiles in the CSFSERV general resource class."

6. Activate logging for CSFSERV using these commands:
   - ALTUSER userid UAUDIT - audits a userid.
   - RALTER class-name profile-name AUDIT(*audit-attempt*[(*audit-access-level*)]) - used by the profile owner.

     RALTER class-name profile-name GLOBALAUDIT(*access-attempt*[(*audit-access-level*)]) - used by a user with AUDITOR authority to set up profiles.
   - SETROPTS CLASSACT(CSFSERV) RACLIST(CSFSERV)
     SETR LOGOPTIONS(CSFSERV(....))

   For more information on RACF RDEFINE, RALTER, and SETR, see the *z/OS Security Server RACF Command Language Reference*.

Setting up profiles in the CSFSERV general resource class

To set up profiles in the CSFSERV general resource class, take these steps.

Note: The CSFSERV class grants access to the service if there is no profile. You can create a global generic profile to restrict access.

1. Define appropriate profiles in the CSFSERV class:

   ```
   RDEFINE  CSFSERV profile-name  UACC(NONE)
            other-optional-operands
   ```

   Where *profile-name* is the profile used to protect the resource. Table 35 lists the resources used by ICSF and PKDS #11 callable services. Table 36 on page 91 shows the resource names used by ICSF TSO panels, utilities, and compatibility services for PCF macros.

   To determine which services are used by PKCS #11 services, see 'Controlling access to tokens' in Chapter 1 of *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*. Users must be SAF authorized to the CSFSERV profile for these service for PKCS #11 services to execute.

Table 35. Resource names for ICSF callable services

Resource name	Callable service names	Callable service description
CSFAPG	CSNBAPG CSNEAPG	Authentication Parameter Generate
CSFCKC	CSNBCKC CSNECKC	CVV Key Combine
CSFCKI	CSNBCKI CSNECKI	Clear Key Import
CSFCKM	CSNBCKM CSNECKM	Multiple Clear Key Import
CSFCPA	CSNBCPA CSNECPA	Clear PIN Generate Alternate
CSFCPE	CSNBCPE CSNECPE	Clear PIN Encrypt
CSFCRC	CSFCRC CSFCRC6	Coordinated KDS Administration
CSFCSG	CSNBCSG CSNECSG	VISA CVV Service Generate
CSFCSV	CSNBCSV CSNECSV	VISA CVV Service Verify
CSFCTT2	CSNBCTT2 CSNECTT2	Ciphertext Translate2
CSFCTT3	CSNBCTT3 CSNECTT3	Ciphertext Translate2 (with ALET)
CSFCVE	CSNBCVE CSNECVE	Cryptographic Variable Encipher
CSFCVT	CSNBCVT CSNECVT	Control Vector Translate
CSFDCM	CSNBDCM CSNEDCM	Derive ICC MK
CSFDCO	CSNBDCO CSNEDCO	Decode
CSFDEC	CSNBDEC CSNEDEC	Decipher

Table 35. Resource names for ICSF callable services (continued)

Resource name	Callable service names	Callable service description
CSFDEC1	CSNBDEC1 CSNEDEC1	Decipher (with ALET)
CSFDKG	CSNBDKG CSNEDKG	Diversified Key Generate
CSFDKG2	CSNBDKG2 CSNEDKG2	Diversified Key Generate2
CSFDKM	CSNBDKM CSNEDKM	Data Key Import
CSFDKX	CSNBDKX CSNEDKX	Data Key Export
CSFDMP	CSNBDMP CSNEDMP	DK Migrate PIN
CSFDPC	CSNBDPC CSNEDPC	DK PIN Change
CSFDPCG	CSNBDPCG CSNEDPCG	DK PRW CMAC Generate
CSFDDPG	CSNBDDPG CSNEDDPG	DK Deterministic PIN Generate
CSFDPMT	CSNBDPMT CSNEDPMT	DK PAN Modify in Transaction
CSFDPNU	CSNBDPNU CSNEDPNU	DK PRW Card Number Update
CSFDPT	CSNBDPT CSNEDPT	DK PAN Translate
CSFDRP	CSNBDRP CSNEDRP	DK Regenerate PRW
CSFDPV	CSNBDPV CSNEDPV	DK PIN Verify
CSFDRPG	CSNBDRPG CSNEDRPG	DK Random PIN Generate
CSFDSG	CSNDDSG CSNFDSG CSFPPS2 CSFPPS26	Digital Signature Generate PKCS #11 Private key structure sign
CSFDSK	CSNBDSK CSNEDSK	Derive Session Key
CSFDSV	CSNDDSV CSNFDSV CSFPPV2 CSFPPV26	Digital Signature Verify PKCS #11 Public key structure verify
CSFEAC	CSNBEAC CSNEEAC	EMV Transaction Service
CSFECO	CSNBECO CSNEECO	Encode
CSFEDH	CSNDEDH CSNFEDH	ECC Diffie-Hellman

Table 35. Resource names for ICSF callable services (continued)

Resource name	Callable service names	Callable service description
CSFENC	CSNBENC CSNEENC	Encipher
CSFENC1	CSNBENC1 CSNEENC1	Encipher (with ALET)
CSFEPG	CSNBEPG CSNEEPG	Encrypted PIN Generate
CSFESC	CSNBESC CSNEESC	EMV Scripting Service
CSFEVF	CSNBEVF CSNEEVF	EMV Verification Functions
CSFFPED	CSNBFPED CSNEFPED	FPE Decipher
CSFFPEE	CSNBFPEE CSNEFPEE	FPE Encipher
CSFFPET	CSNBFPET CSNEFPET	FPE Translate
CSFGIM	CSNBGIM CSNEGIM	Generate Issuer MK
CSFHMG	CSNBHMG CSNEHMG	HMAC Generate
CSFHMG1	CSNBHMG1 CSNEHMG1	HMAC Generate (with ALET)
CSFHMV	CSNBHMV CSNEHMV	HMAC Verify
CSFHMV1	CSNBHMV1 CSNEHMV1	HMAC Verify (with ALET)
CSFIQA	CSFIQA CSFIQA6	ICSF Query Algorithm
CSFIQF	CSFIQF CSFIQF6	ICSF Query Facility
CSFKDSL[3]	CSFKDSL CSFKDSL6	Key Data Set List
CSFKDMR[3]	CSFKDMR CSFKDMR6	Key Data Set Metadata Read
CSFKDMW[3]	CSFKDMW CSFKDMW6	Key Data Set Metadata Write
CSFKDU[3, 4]	CSFKDU CSFKDU6	Key Dataset Update It is recommended that this profile is defined with UACC(NONE).
CSFKET	CSNBKET CSNEKET	Key Encryption Translate
CSFKEX	CSNBKEX CSNEKEX	Key Export
CSFKGN	CSNBKGN CSNEKGN	Key Generate

Table 35. Resource names for ICSF callable services (continued)

Resource name	Callable service names	Callable service description
CSFKGN2	CSNBKGN2 CSNEKGN2	Key Generate2
CSFKIM	CSNBKIM CSNEKIM	Key Import
CSFKPI	CSNBKPI CSNEKPI	Key Part Import
CSFKPI2	CSNBKPI2 CSNEKPI2	Key Part Import2
CSFKRC	CSNBKRC CSNEKRC	Key Record Create
CSFKRC2	CSNBKRC2 CSNEKRC2	Key Record Create2
CSFKRD	CSNBKRD CSNEKRD	Key Record Delete
CSFKRR	CSNBKRR CSNEKRR	Key Record Read
CSFKRR2	CSNBKRR2 CSNEKRR2	Key Record Read2
CSFKRW	CSNBKRW CSNEKRW	Key Record Write
CSFKRW2	CSNBKRW2 CSNEKRW2	Key Record Write2
CSFKTR	CSNBKTR CSNEKTR	Key Translate
CSFKTR2	CSNBKTR2 CSNEKTR2	Key Translate2
CSFKYT	CSNBKYT CSNEKYT	Key Test
CSFKYT2	CSNBKYT2 CSNEKYT2	Key Test2
CSFKYTX	CSNBKYTX CSNEKYTX	Key Test Extended
CSFMDG	CSNBMDG CSNEMDG	MDC Generate
CSFMDG1	CSNBMDG1 CSNEMDG1	MDC Generate (with ALET)
CSFMGN	CSNBMGN CSNEMGN	MAC Generate
CSFMGN1	CSNBMGN1 CSNEMGN1	MAC Generate (with ALET)
CSFMGN2	CSNBMGN2 CSNEMGN2	MAC Generate2
CSFMGN3	CSNBMGN3 CSNEMGN3	MAC Generate2 (with ALET)
CSFMPS	CSFMPS CSFMPS6	ICSF Multi-Purpose Service

Table 35. Resource names for ICSF callable services (continued)

Resource name	Callable service names	Callable service description
CSFMVR	CSNBMVR CSNEMVR	MAC Verify
CSFMVR1	CSNBMVR1 CSNEMVR1	MAC Verify (with ALET)
CSFMVR2	CSNBMVR2 CSNEMVR2	MAC Verify2
CSFMVR3	CSNBMVR3 CSNEMVR3	MAC Verify2 (with ALET)
CSFOWH[1]	CSNBOWH CSNEOWH CSFPOWH CSFPOWH6	One-Way Hash Generate and PKCS #11 One-way hash, sign, or verify
CSFOWH1 [1]	CSNBOWH1 CSNEOWH1	One-Way Hash Generate (with ALET)
CSFPCI	CSFPCI CSFPCI6	PCI Interface Callable Service
CSFPCU	CSNBPCU CSNEPCU	PIN Change/Unblock
CSFPEX	CSNBPEX CSNEPEX	Prohibit Export
CSFPEXX	CSNBPEXX CSNEPEXX	Prohibit Export Extended
CSFPFO	CSNBPFO CSNEPFO	Recover PIN From Offset
CSFPGN	CSNBPGN CSNEPGN	Clear PIN Generate
CSFPKD	CSNDPKD CSNFPKD CSFPPD2 CSFPPD26	PKA Decrypt PKCS #11 Private key structure decrypt
CSFPKE	CSNDPKE CSNFPKE CSFPPE2 CSFPPE26	PKA Encrypt PKCS #11 Public key structure encrypt
CSFPKG	CSNDPKG CSNFPKG	PKA Key Generate
CSFPKI	CSNDPKI CSNFPKI	PKA Key Import
CSFPKRC	CSNDKRC CSNFKRC	PKDS Record Create
CSFPKRD	CSNDKRD CSNFKRD	PKDS Record Delete
CSFPKRR	CSNDKRR CSNFKRR	PKDS Record Read
CSFPKRW	CSNDKRW CSNFKRW	PKDS Record Write

Table 35. Resource names for ICSF callable services (continued)

Resource name	Callable service names	Callable service description
CSFPKT	CSNDPKT CSNFPKT	PKA Key Translate
CSFPKTC	CSNDKTC CSNFKTC	PKA Key Token Change
CSFPKX	CSNDPKX CSNFPKX	PKA Public Key Extract
CSFPRR2	CSNDKRR2 CSNFKRR2	PKDS Key Record Read2
CSFPTR	CSNBPTR CSNEPTR	Encrypted PIN Translate
CSFPTRE	CSNBPTRE CSNEPTRE	Encrypted PIN Translate Enhanced
CSFPVR	CSNBPVR CSNEPVR	Encrypted PIN Verify
CSFRKA	CSNBRKA CSNERKA	Restrict Key Attribute
CSFRKD	CSNDRKD CSNFRKD	Retained Key Delete
CSFRKL	CSNDRKL CSNFRKL	Retained Key List
CSFRKX	CSNDRKX CSNFRKX	Remote Key Export
CSFRNG[2]	CSNBRNG CSNERNG CSFPPRF CSFPPRF6	Random Number Generate (returning an 8-byte random number) and PKCS #11 Pseudo-random function
CSFRNGL[2]	CSNBRNGL CSNERNGL	Random Number Generate (returning a random number of a length that is specified by the caller)
CSFRRT [3, 4]	CSFRRT CSFRRT6	Key Dataset Record Retrieve It is recommended that this profile is defined with UACC(NONE) and no user be given access as it is for diagnostic purposes only.
CSFSAD	CSNBSAD CSNESAD	Symmetric Algorithm Decipher
CSFSAD1	CSNBSAD1 CSNESAD1	Symmetric Algorithm Decipher (with ALET)
CSFSAE	CSNBSAE CSNESAE	Symmetric Algorithm Encipher
CSFSAE1	CSNBSAE1 CSNESAE1	Symmetric Algorithm Encipher (with ALET)
CSFSBC	CSNDSBC CSNFSBC	SET Block Compose
CSFSBD	CSNDSBD CSNFSBD	SET Block Decompose
CSFSKI	CSNBSKI CSNESKI	Secure Key Import

Table 35. Resource names for ICSF callable services (continued)

Resource name	Callable service names	Callable service description
CSFSKI2	CSNBSKI2 CSNESKI2	Secure Key Import2
CSFSKM	CSNBSKM CSNESKM	Multiple Secure Key Import
CSFSKY	CSNBSKY CSNESKY	Secure Messaging for Keys
CSFSPN	CSNBSPN CSNESPN	Secure Messaging for PINs
CSFSXD	CSNDSXD CSNFSXD	Symmetric Key Export with Data
CSFSYG	CSNDSYG CSNFSYG	Symmetric Key Generate
CSFSYI	CSNDSYI CSNFSYI	Symmetric Key Import
CSFSYI2	CSNDSYI2 CSNFSYI2	Symmetric Key Import2
CSFSYX	CSNDSYX CSNFSYX	Symmetric Key Export
CSFTBC	CSNDTBC CSNFTBC	Trusted Block Create
CSFTRV	CSNBTRV CSNETRV	Transaction Validation
CSFT31I	CSNBT31I CSNET31I	TR-31 Import
CSFT31X	CSNBT31X CSNET31X	TR-31 Export
CSFUKD	CSNBUKD CSNEUKD	Unique Key Derive
CSFWRP	CSFWRP CSFWRP6	Key Token Wrap
CSF1DVK	CSFPDVK CSFPDVK6	PKCS #11 Derive key
CSF1DMK	CSFPDMK CSFPDMK6	PKCS #11 Derive multiple keys
CSF1HMG	CSFPHMG CSFPHMG6	PKCS #11 Generate MAC
CSF1GKP	CSFPGKP CSFPGKP6	PKCS #11 Generate key pair
CSF1GSK	CSFPGSK CSFPGSK6	PKCS #11 Generate secret key
CSF1GAV	CSFPGAV CSFPGAV6	PKCS #11 Get attribute value
CSF1PKS	CSFPPKS CSFPPKS6	PKCS #11 Private key sign
CSF1PKV	CSFPPKV CSFPPKV6	PKCS #11 Public key verify

Table 35. Resource names for ICSF callable services (continued)

Resource name	Callable service names	Callable service description
CSF1SKD	CSFPSKD CSFPSKD6	PKCS #11 Secret key decrypt
CSF1SKE	CSFPSKE CSFPSKE6	PKCS #11 Secret key encrypt
CSF1SAV	CSFPSAV CSFPSAV6	PKCS #11 Set attribute value
CSF1TRC	CSFPTRC CSFPTRC6	PKCS #11 Token record create
CSF1TRD	CSFPTRD CSFPTRD6	PKCS #11 Token record delete
CSF1TRL	CSFPTRL CSFPTRL6	PKCS #11 Token record list
CSF1UWK	CSFPUWK CSFPUWK6	PKCS #11 Unwrap key
CSF1HMV	CSFPHMV CSFPHMV6	PKCS #11 Verify MAC
CSF1WPK	CSFPWPK CSFPWPK6	PKCS #11 Wrap key

[1] If the CSF.CSFSERV.AUTH.CSFOWH.DISABLE resource is defined within the XFACILIT class, the SAF authorization check is disabled for this resource. Disabling the SAF check might improve the performance of your applications.

[2] If the CSF.CSFSERV.AUTH.CSFRNG.DISABLE resource is defined within the XFACILIT class, the SAF authorization check is disabled for this resource. Disabling the SAF check might improve the performance of your application.

[3] These services do not perform SAF authorization checks against key labels or handles (SAF classes CSFKEYS and CRYPTOZ). Therefore, any user ID that is permitted to use these services is able to access any KDS record. The level of access (read or update) depends on the operation of the service.

[4] Access to these services is denied if there is no covering profile in the CSFSERV class.

Table 36. Resource names for ICSF TSO panels, utilities, and compatibility services for PCF macros

Resource Name	Utility and Callable Service Description
CSFCMK	Change master key utility, including the panel for a local change master key, the Coordinated KDS Administration service, and CSFEUTIL.
CSFCONV	PCF CKDS to ICSF CKDS conversion utility
CSFCRC	Coordinated KDS Administration
CSFDKCS	Master key entry utility
CSFEDC	Compatibility service for the PCF CIPHER macro
CSFEMK	Compatibility service for the PCF EMK macro
CSFGKC	Compatibility service for the PCF GENKEY macro
CSFGKF	Generate key fingerprint. Required by KGUP if key lifecycle auditing is enabled.
CSFKGUP	Key generation utility program

Table 36. Resource names for ICSF TSO panels, utilities, and compatibility services for PCF macros (continued)

Resource Name	Utility and Callable Service Description
CSFOPKL	Operational key load
CSFPCAD	Cryptographic processors management (activate/deactivate)
CSFPKDR	PKDS reencipher and PKDS refresh utilities
CSFPMCI	Pass phrase master key/KDS initialization utility
CSFREFR	Refresh CKDS or PKDS utility, including the panels for a local refresh, the Coordinated KDS Administration service, and CSFEUTIL (CKDS) and CSFPUTIL (PKDS).
CSFRENC	Reencipher CKDS or PKDS utility, including the panels for a local refresh, the Coordinated KDS Administration service, and CSFEUTIL (CKDS) and CSFPUTIL (PKDS).
CSFRSWS	Administrative control functions utility (ENABLE)
CSFRWP	CKDS Conversion2 - rewrap option.
CSFRTC	Compatibility service for the CUSP or PCF RETKEY macro
CSFSMK	Set master key utility
CSFSSWS	Administrative control functions utility (DISABLE)
CSFUDM	User Defined Extensions (UDX) management functions

Note:

a. As with any RACF general resource profile, if you want to change the profile later, use the RALTER command. To change the access list, use the PERMIT command as described in the next step.
b. If you have already started ICSF, you need to refresh the in-storage profiles. See Step 3.
c. You can specify other operands, such as auditing (AUDIT operand), on the RDEFINE or RALTER commands.
d. If the security administrator has activated generic profile checking for the CSFSERV class, you can create generic profiles that use the generic characters * and %. This is the same as with any RACF general resource class.

For example, if generic profile checking is in effect, these profiles enable you to specify which users and jobs can use the Ciphertext Translate callable services. No other services can be used by any job on the system.

```
RDEFINE  CSFSERV  CSFCTT2  UACC(NONE)
RDEFINE  CSFSERV  CSFCTT3  UACC(NONE)
RDEFINE  CSFSERV  *        UACC(NONE)
```

2. Give appropriate users (preferably groups) access to the profiles:

```
PERMIT  profile-name  CLASS(CSFSERV) ID(groupid)  ACCESS(READ)
```

3. When the profiles are ready to be used, ask the security administrator to activate the CSFSERV class and refresh the in-storage RACF profiles:

```
SETROPTS  CLASSACT(CSFSERV)
SETROPTS RACLIST(CSFSERV) REFRESH
```

Setting up profiles in the CSFKEYS general resource class

For setting up profiles in the CRYPTOZ class for PKCS #11 tokens and objects, see 'Controlling access to tokens' in Chapter 1 of *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

To set up profiles in the CSFKEYS general resource class, take these steps.

Note: The CSFKEYS class grants access to the key if there is no profile. You can create a global generic profile to restrict access.

1. Define appropriate profiles in the CSFKEYS class:

   ```
   RDEFINE  CSFKEYS label  UACC(NONE)
            other-optional-operands
   ```

   where *label* is the label by which the key is defined in the CKDS or PKDS. Note that if an application uses a token instead of a key label, no authorization checking is done on the use of the key.

   Note:

   a. As with any SAF profile, if you want to change the profile later, use the RALTER command. To change the access list, use the PERMIT command as described in the next step.
   b. If you have already started ICSF, you need to refresh the in-storage profiles. See Step 3.
   c. You can specify other operands, such as auditing (AUDIT operand), on the RDEFINE or RALTER commands.
   d. If the security administrator has activated generic profile checking for the CSFKEYS class, you can create generic profiles using the generic characters * and %. This is the same as any SAF general resource class.
2. Give appropriate users (preferably groups) access to the profiles:

   ```
   PERMIT  profile-name  CLASS(CSFKEYS)
           ID(groupid)  ACCESS(READ)
   ```

   Notes:

   - READ authority is the default authority for access to PKDS and CKDS labels for all usage. See “Increasing the level of authority needed to modify key labels” on page 46 for controls available to increase the authority for certain usages.
   - For the exclusive purpose of requiring UPDATE instead of READ authority when transferring a secure symmetric key from encryption under the master key to encryption under an RSA key, you can define profiles in the XCSFKEY class. Profiles in the XCSFKEY class are used in authorization checks only when the Symmetric Key Export service (CSNDSYX, CSNFSYX, or CSNDSXD) is called. See “Increasing the level of authority required to export symmetric keys” on page 48 for additional information.
3. When the profiles are ready to be used, ask the security administrator to activate the CSFKEYS class and refresh the in-storage SAF profiles:

   ```
   SETROPTS  CLASSACT(CSFKEYS)
   SETROPTS RACLIST(CSFKEYS) REFRESH
   ```

Enabling use of encrypted keys in callable services that exploit CPACF

The Field Level Encipher, Field Level Decipher, Symmetric Key Encipher, and Symmetric Key Decipher callable services exploit CP Assist for Cryptographic Functions (CPACF) for improved performance.

The CKDS Key Record Read2 callable service can return the protected-key CPACF form of the CCA token to a caller with sufficient authority (either system key or supervisor state).

These services support encrypted AES and DES key tokens via the key label. This support requires the Crypto Express3 or later feature. The encrypted keys tokens must be stored in the CKDS and have a CSFKEYS profile with the ICSF segment.

A CSFKEYS profile can contain an ICSF segment, which specifies rules for key use. The SYMCPACFWRAP field of the ICSF segment enables you to specify whether ICSF can rewrap the encrypted key using the CPACF wrapping key. The specification:

- SYMCPACFWRAP(YES) indicates that encrypted keys covered by the profile can be rewrapped.
- SYMCPACFWRAP(NO), which is the default, indicates that encrypted keys covered by the profile cannot be rewrapped.

For all five services to utilize protected keys, SYMCPACFWRAP(YES) must be enabled in the covering profile.

For CKDS Key Record Read2, the SYMCPACFRET field of the ICSF segment enables you to specify whether ICSF can return the protected-key form of the CCA token to a caller. The specification:

- SYMCPACFRET(YES) indicates that keys covered by the profile can be returned to the caller in their protected-key CPACF form.
- SYMCPACFRET(NO), which is the default, indicates that keys covered by the profile cannot be returned to the caller in their protected-key CPACF form.

Rewrapping the encrypted key using the CPACF wrapping key is necessary in order to use an encrypted key as input to the Symmetric Key Encipher or Symmetric Key Decipher callable services. You should be aware, however, that although the rewrapping operation ensures that no key is visible in application or system storage, the operation also requires the key to exist in the clear outside of the tamper-resistant hardware boundary. If your installation requires that a particular encrypted key must never exist outside of the tamper-resistant hardware boundary, do not use the SYMCPACFWRAP(YES) specification in a CSFKEYS profile that covers the key.

For example, say the CSFKEYS general resource profile DES.CHAOS.CAT covers an encrypted key stored in the CKDS that you would like to use as input to the Symmetric Key Encipher and Symmetric Key Decipher callable services. The following command modifies the SYMCPACFWRAP field of the profile's ICSF segment to allow this. The SETROPTS RACLIST command is used to refresh the CSFKEYS class in common storage.

```
RALTER CSFKEYS DES.CHAOS.CAT ICSF(SYMCPACFWRAP(YES))
SETROPTS RACLIST(CSFKEYS) REFRESH
```

Chapter 6. Using the pass phrase initialization utility

The pass phrase initialization utility allows the casual user of ICSF to install the necessary master keys on the cryptographic coprocessors, and initialize the CKDS and PKDS with a minimal effort. This topic describes how to use this utility to get up and running quickly.

Note: The pass phrase initialization utility is used to install the master keys for CCA coprocessors only. The master key for Enterprise PKCS #11 coprocessors can only be entered via a TKE workstation as explained in Chapter 8, "Managing PKCS #11 master keys," on page 159.

The pass phrase is case sensitive and should be chosen according to these rules:
- It can contain a minimum of 16 and a maximum of 64 characters.
- It can include any characters in the EBCDIC character set.
- It can contain imbedded blanks, but leading and trailing blanks are truncated.

Important: The same pass phrase will always produce the same master key values, and is therefore as critical and sensitive as the master key values themselves. Make sure you save the pass phrase so that you can later reenter it if needed (for example, if you need to restore master key values that have been cleared). Because of the sensitive nature of the pass phrase, make sure you secure it in a safe place.

The pass phrase initialization utility can:
- Initialize a system for the first time (Initialize system).
- Reinitialize a system where the master keys have been cleared (Reinitialize system).
- Initialize a new system when migrating to a new server with an existing CKDS and PKDS (Reinitialize system).
- Load the master keys on CCA coprocessors that are brought online after system initialization (Add coprocessors).
- Add new master keys to the system after migrating to a newer server (Add AES-MK or Add missing MKs).

You cannot use this utility to change master keys. To change master keys you need to use either the master key entry panels or the TKE workstation.

If you plan on sharing your CKDS or PKDS within your sysplex, refer to Chapter 10, "Running in a Sysplex Environment," on page 171 for important information.

Starting with ICSF FMID HCR77A0, the DES master key may be 16 or 24 bytes long. If the **DES master key – 24-byte key** access control point is enabled, the pass phrase initialization utility will load a 24-byte value to the DES master key. A TKE workstation is required to enable access control points.

Requirements for running the Pass Phrase Initialization Utility

When you run the pass phrase initialization utility for the first time, you must perform these steps:

1. Install the ICSF program product according to the instructions in *z/OS Planning for Installation*.
2. Create an empty CKDS.
3. Create an empty PKDS.
4. Create an installation options data set.
5. Create an ICSF startup procedure.
6. Ensure ICSF is running in COMPAT(NO) mode
7. Start ICSF.
8. Access the ICSF panels.

These steps are described in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

There are three formats of the CKDS. These are described in *z/OS Cryptographic Services ICSF System Programmer's Guide*:
- Fixed-length record – LRECL 252 (supported by all releases of ICSF).
- Variable-length record – LRECL 1024 (supported by ICSF FMID HCR7780 and later releases).
- Common record format (KDSR) – LRECL 2048 (supported by ICSF FMID HCR77A1 and later releases).

There are two formats for the PKDS. These are described in *z/OS Cryptographic Services ICSF System Programmer's Guide*. The LRECL is the same for both formats, but the internal record structure is different.
- Base format (supported by all releases of ICSF).
- Common record format (KDSR) (supported by ICSF FMID HCR77A1 and later releases).

The pass phrase initialization utility can be used with all formats of CKDS and PKDS. If you want to use the KDSR format for the CKDS and PKDS, select KDSR format on the PPINIT panel when initializing your system.

SAF Protection

The pass phrase initialization utility is primarily protected by the CSFPMCI profile for the CSFSERV SAF class. Only the users authorized to the CSFPMCI profile can use the utility. In addition, the user must be authorized to all or some of these services which are used by the utility. The services used are dependent on the function or functions of PPINIT that are being utilized.
- CSFOWH
- CSFDKCS
- CSFCMK
- CSFECO
- CSFMDG
- CSFSMK
- CSFREFR
- CSFRSWS

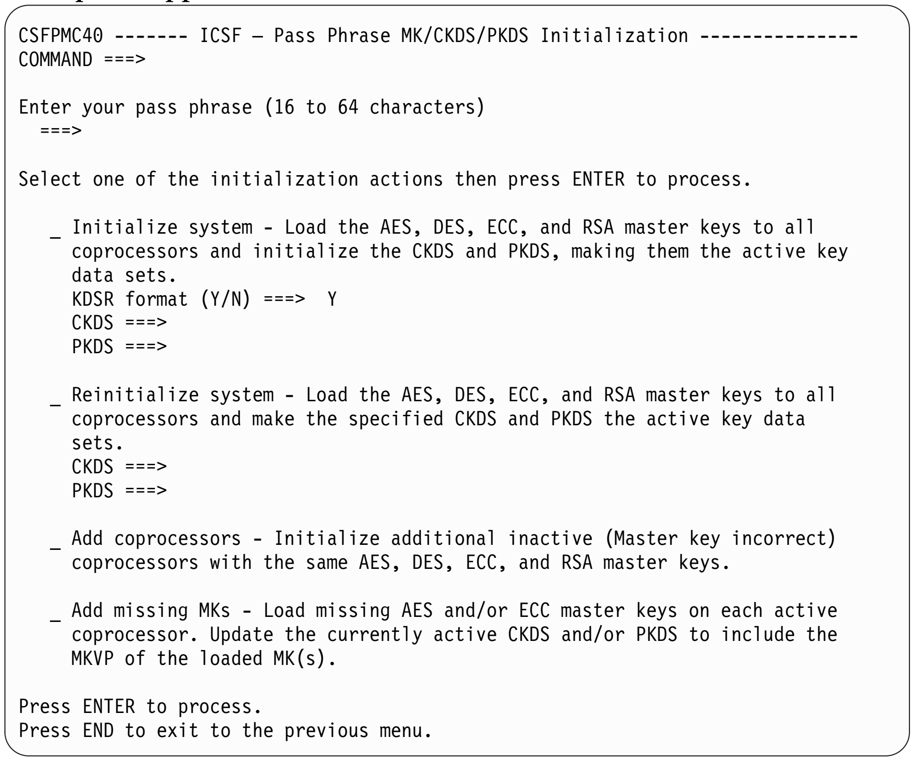
Running the Pass Phrase Initialization Utility

When you start ICSF, you use the ICSF panels to run the pass phrase initialization utility. When you access the ICSF panels, the primary menu panel appears. The ICSF FMID appears in the upper left corner (it toggles to the panel identification ID).

Select option 6, PPINIT, and press ENTER to begin the pass phrase initialization utility. The pass phrase panel appropriate for your hardware configuration appears.

If you are running on:

z196, z114, or later zHardware and with a CEX3C or later, you have AES and ECC master key support and the CSFPMC40 Pass Phrase MK/CKDS/PKDS Initialization panel appears.



```
CSFPMC40 ------- ICSF – Pass Phrase MK/CKDS/PKDS Initialization ---------------
COMMAND ===>

Enter your pass phrase (16 to 64 characters)
  ===>

Select one of the initialization actions then press ENTER to process.

   _ Initialize system - Load the AES, DES, ECC, and RSA master keys to all
     coprocessors and initialize the CKDS and PKDS, making them the active key
     data sets.
     KDSR format (Y/N) ===>  Y
     CKDS ===>
     PKDS ===>

   _ Reinitialize system - Load the AES, DES, ECC, and RSA master keys to all
     coprocessors and make the specified CKDS and PKDS the active key data
     sets.
     CKDS ===>
     PKDS ===>

   _ Add coprocessors - Initialize additional inactive (Master key incorrect)
     coprocessors with the same AES, DES, ECC, and RSA master keys.

   _ Add missing MKs - Load missing AES and/or ECC master keys on each active
     coprocessor. Update the currently active CKDS and/or PKDS to include the
     MKVP of the loaded MK(s).

Press ENTER to process.
Press END to exit to the previous menu.
```

z9, z10, or later with the Nov. 2008 or later licensed internal code (LIC), you have AES master key support and the CSFPMC30 Pass Phrase MK/CKDS/PKDS Initialization panel appears.

```
CSFPMC30 ------- ICSF - Pass Phrase MK/CKDS/PKDS Initialization ---
 Command ===>
 Enter your pass phrase (16 to 64 characters)
   ===>

 Select one of the initialization actions then press ENTER to process.

_ Initialize system - Load the AES, DES and RSA master keys to all
    coprocessors and initialize the CKDS and PKDS,
    making them the active key data sets.
    KDSR format (Y/N) ===>  Y
    CKDS ===>
    PKDS ===>

_ Reinitialize system - Load the AES, DES, and RSA master keys to all
  coprocessors and make the specified CKDS and PKDS the active key data sets.
    CKDS ===>
    PKDS ===>


_ Add coprocessors - Initialize additional inactive (Master key incorrect)
   coprocessors with the same AES, DES, and RSA master keys.

_ Add AES-MK - Add the AES master key to all active coprocessors and the
   current CKDS.


 Press ENTER to process.
 Press END   to exit to the previous menu.
```

The CSFPMC10 Pass Phrase MK/CKDS/PKDS Initialization panel appears.

```
CSFPMC10 ------- ICSF - Pass Phrase MK/CKDS/PKDS Initialization ---
 Command ===>
 Enter your pass phrase (16 to 64 characters)
   ===>

 Select one of the initialization actions then press ENTER to process.

_ Initialize system - Load the DES and RSA master keys to all
    coprocesors and initialize the CKDS and the PKDS,
    making them the active key data sets.
    KDSR format (Y/N) ===>  Y
    CKDS ===>
    PKDS ===>

_ Reinitialize system - Load the DES and RSA master keys to all
    coprocesors and make the specified CKDS and the PKDS the active key data
    sets.
    CKDS ===>
    PKDS ===>


 _ Add coprocessors - Initialize additional inactive (Master key incorrect)
   coprocessors with the same DES and RSA master keys.


 Press ENTER to process.
 Press END   to exit to the previous menu.
```

This utility uses the pass phrase, a series of constants, and the MD5 and SHA-256 hash functions to calculate the value of the master keys. For details on how the values of master keys are calculated, see "Pass Phrase Initialization master key calculations" on page 399.

Steps for initializing a system for the first time

Use this section when starting ICSF for the first time to load the master keys and to initialize your CKDS and PKDS.

1. Type the pass phrase in the space provided and select the 'Initialize system' option by placing an “s” in the “_” field.

   Note: Make sure you save the pass phrase and store it in a secure location. The same pass phrase will always produce the same master key values and is therefore as critical and sensitive as the master key values themselves. Make sure you save the pass phrase so that you can later reenter it if needed (for example, if you need to restore master key values that have been cleared). Because of the sensitive nature of the pass phrase, make sure you secure it in a safe place.
2. Fill in the CKDS and PKDS fields with the names of two VSAM data sets that have not been initialized.

   If you want to use the KDSR format for the PKDS, you must answer “Y” to KDSR format. The format of the CKDS will be determined from the data set attributes.
3. Press ENTER to run the utility.

   This utility calculates the value of the master keys, loads the master keys on all coprocessors, and initializes the CKDS and PKDS.

   Messages on the bottom half of the panel display the progress of the utility.
4. If there are any valid master keys on your system, the “CSFPMC20 — Pass Phrase MK/CKDS/PKDS Initialization” on page 392 panel appears. This prevents you from making a mistake and changing a system that is already operational.
5. When the utility has completed successfully, press END to return to the primary menu.

   If the initialization was not successful, you will have to reallocate the CKDS and PKDS again before retrying.

Steps for reinitializing a system

Use this section when you are reinitializing a system after the master keys have been cleared or you are migrating to a new server.

You must use the same pass phrase you originally used to initialize the master keys, CKDS and PKDS.

Note: If the system supports any additional master key type and there is no MKVP in the CKDS or PKDS for the key type, the master key will not become active. Use the 'Add missing MKs' option after the system has been initialized to add any new master keys that the new server supports.

1. Type the pass phrase in the space that is provided and select the 'Reinitialize system' option by placing an “s” in the “_” field.
2. Fill in the CKDS and PKDS fields with the names of VSAM data sets that were used when your system was initialized.
3. Press ENTER to run the utility.

   The utility verifies that the pass phrase matches the CKDS and PKDS. The master key values are calculated and loaded into all coprocessors.

   Messages on the bottom half of the panel display the progress of the utility.

4. If there are any valid master keys on your system, the “CSFPMC20 — Pass Phrase MK/CKDS/PKDS Initialization” on page 392 panel appears. This prevents you from making a mistake and changing a system that is already operational.
5. When the utility has completed successfully, press END to return to the primary menu.

Steps for adding a CCA coprocessor after first time Pass Phrase Initialization

Use this section when you add more coprocessors to your system. ICSF will load the same master key values into the new coprocessors.

You must use the same pass phrase as when you originally ran PPINIT to initialize the active CKDS and PKDS.

Note: At least one coprocessor must be active and the PKA callable services control must be enabled where applicable.

1. Type the pass phrase in the space that is provided and select the 'Add coprocessors' option by placing an “s” in the “_” field.
2. Press ENTER to run the utility.

   The utility will calculate the master key values and load the master keys on all coprocessors that are not active.

   Messages on the bottom half of the panel display the progress of the utility.
3. When the utility has completed successfully, press END to return to the primary menu.

Steps to add missing master keys

Use this section to add new master keys after migrating to a server which supports master keys that were not available when your CKDS and PKDS were initialized. The master key value will be calculated from your pass phrase, loaded and set. The CKDS and PKDS will be updated so keys for the new algorithms can be stored in them.

You must use the same pass phrase you originally used to initialize the master keys, CKDS and PKDS.

1. Type the pass phrase in the space that is provided and select the 'Add missing MKs' option by placing an 's' in the '_' field.
2. Press ENTER to run the utility.

   The utility will verify the pass phrase matches the active CKDS and PKDS. The values of the new master keys are calculated and loaded into all coprocessors. The CKDS and PKDS are updated with the master key verification pattern of the new master keys.

   Messages on the bottom half of the panel display the progress of the utility.
3. When the utility has completed successfully, press END to return to the primary menu.

If you have multiple systems and are adding missing master keys, you will follow the step listed above on one system. For the rest of the systems, enter your pass phrase, the names of the initialized CKDS and PKDS, select 'Reinitialize system', and then press ENTER to run the utility.

Initializing multiple systems with pass phrase initialization utility

Use this scenario when using the pass phrase initialization utility to initialize more than one system where the CKDS and PKDS will be shared by all systems:

1. Select a system, A. This system will be used to initialize the CKDS and PKDS.
2. On system A, enter your pass phrase, the names of the empty CKDS and PKDS and select 'Initialize system' and press ENTER to run the utility.
3. When system A has been successfully initialized, the rest of the systems to share the CKDS and PKDS can be initialized.
4. For the rest of the systems, enter your pass phrase, the names of the initialized CKDS and PKDS and select 'Reinitialize system' and press ENTER to run the utility.

Chapter 7. Managing CCA Master Keys

This topic describes how to manage the CCA master keys, the Cryptographic Key Data Set (CKDS), and the Public Key Data Set (PKDS).

This topic describes how to:
- Enter master keys using the Master Key Entry panels.
- Reenter master keys when they are cleared.
- Initialize the CKDS and PKDS for the first time.
- Update the CKDS or PKDS with an additional master key.
- Refresh the CKDS and PKDS (load a different KDS or reload the same KDS).
- Change master keys including reenciphering the KDS.
- Add coprocessors after initialization.
- Clear master keys.

The CCA master keys are described in "Master keys" on page 11.

Introduction

Master keys are required for cryptographic operations. They can be loaded by using the TKE workstation or the ICSF Master Key Entry panels. Either method is acceptable anywhere that the description says to load the master keys:
- See "Entering master key parts" on page 111 for information on using the ICSF Master Key Entry panels to enter master key parts.
- See *z/OS Cryptographic Services ICSF TKE Workstation User's Guide* for information on using the TKE workstation to enter master key parts.

Important: Regardless of the source of the master key parts or the method of key entry, **make sure that the key parts are recorded and saved in a secure location.** When you are entering the key parts for the first time, be aware that **you might need to reenter these same key values later** to restore master key values that have been cleared or for new coprocessors added to your systems.

All master keys are optional. You should load the master keys that are required for your cryptographic applications. You can add the master keys that are not in use at any time.

The CKDS and PKDS are used to store cryptographic keys for use by ICSF callable services. The CKDS and PKDS must be initialized before they can be used. The initialization creates a header record with the verification pattern of the master keys. The process of initializing the key data sets causes the master keys to be set and the coprocessors become active. See "Initializing the key data sets at first-time startup" on page 128.

When you want to start using a new master key after the key data sets have been initialized, the key data sets must be updated to add the master key verification pattern to the header record. See "Updating the key data sets with additional master keys" on page 132 for additional information.

The CKDS and PKDS can be refreshed. You can load the same key data set or a different key data set. Refreshing the KDS does not disrupt cryptographic functions. See "Refreshing the key data sets" on page 134 for additional information.

Master keys should be changed periodically. Changing the master keys involves loading the new master key registers, reenciphering the key data sets, and setting the master keys. The change master key utilities promote the master keys from the new register to the current register and activate the reenciphered key data sets for continuous operation. See "Changing the master keys" on page 140 for additional information.

Important: The master keys are loaded into the new master key register. The utilities that are described in this topic require the master keys to be in the new master key register and not set. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. This option must not be used for a coprocessor or domain that is visible to ICSF. Setting the master keys from the TKE workstation might result in the loss of the cryptographic function of that coprocessor or domain. Setting the master keys should be done by using the ICSF utilities so that ICSF knows that the master keys have changed.

New coprocessors can be added to your systems. For the coprocessors to become active, the master keys must be loaded and set. You need the same master key parts for the current master keys. See "Reentering master keys when they have been cleared" on page 126 for additional information.

Coordinated and local utilities

The utilities that are described in this topic can be used on a single LPAR (local) or for all members of a sysplex sharing a key data set (coordinated). Not all local utilities have a coordinated version.

The coordinated administration utilities, refresh key data set, and master key change operate across sysplex members sharing the same active key data set. The coordinated administration functions simplify master key management by automating the manual process for performing local refreshes and local master key changes. Although a sysplex environment is not required to use these functions, sysplex environments gain the maximum benefit from them when the changes are coordinated across all LPARs sharing the same active KDS.

Coordinated KDS change master key offers further advantages in a sysplex environment. Specifically, a master key change that is initiated from one ICSF instance in the sysplex changes the master key or keys for all ICSF instances in the sysplex that share the same active KDS. The coordinated KDS change master key utility is initiated from a single ICSF instance. This can be either a stand-alone system or a member of a sysplex cluster.

The Master Key Entry panels can be used to load master keys on a single LPAR. See "Entering master key parts" on page 111. The TKE workstation can be used to load master keys on multiple coprocessors for a single LPAR or all coprocessors being administrated by the TKE workstation. For information on using the TKE workstation, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Initialization of the key data sets is done on a single LPAR. See "Initializing the key data sets at first-time startup" on page 128. The initialized key data sets are then used to set the master keys for the remaining LPARs of a sysplex. See Chapter 10, “Running in a Sysplex Environment,” on page 171 for the procedure to initialize a key data set in a sysplex.

Adding a new master key to an existing key data set is done on a single LPAR. See “Updating the key data sets with additional master keys” on page 132 for additional information. The updated key data set is then used to set the new master key for the remaining LPARs of a sysplex. See Chapter 10, “Running in a Sysplex Environment,” on page 171 for the procedure for updating a key data set in a sysplex.

Refreshing a key data set can be done on a single LPAR or for all members sharing the key data set in a sysplex.

- A local CKDS refresh can be done by using the ICSF Key Data Set Management panels (see “Refreshing the key data sets” on page 134) or as a batch job by using CSFEUTIL (see Chapter 17, “Using the ICSF Utility Program CSFEUTIL,” on page 343.
- A local PKDS refresh can be done by using the ICSF Key Data Set Management panels (see “Refreshing the key data sets” on page 134) or as a batch job by using CSFPUTIL (see Chapter 18, “Using the ICSF Utility Program CSFPUTIL,” on page 353).
- A coordinated CKDS or PKDS refresh can be done by using the ICSF Key Data Set Management panels or by writing an application to start the Coordinated KDS Administration (CSFCRC) callable service. For the description of the service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*z/OS Cryptographic Services ICSF Application Programmer's Guide.

Changing the master keys can be done on a single LPAR or for all members sharing the key data set in a sysplex.

- The master key registers are loaded by using the Master Key Entry panels or the TKE workstation.
- A local CKDS reencipher can be done by using the ICSF CKDS Management panels (see “Changing the master keys” on page 140) or as a batch job by using CSFEUTIL (see Chapter 17, “Using the ICSF Utility Program CSFEUTIL,” on page 343).
- A local change symmetric master key can be done by using the ICSF CKDS Management panels (see “Changing the master keys” on page 140) or as a batch job by using CSFEUTIL (see Chapter 17, “Using the ICSF Utility Program CSFEUTIL,” on page 343).
- A local PKDS reencipher can be done by using the ICSF PKDS Management panels (see “Changing the master keys” on page 140) or as a batch job by using CSFPUTIL (see Chapter 18, “Using the ICSF Utility Program CSFPUTIL,” on page 353).
- A local change asymmetric master key can be done by using the ICSF PKDS Management panels. See “Changing the master keys” on page 140.
- A coordinated CKDS or PKDS change master key combines the reenciphering of the key data set and the setting of the master keys. A CKDS or PKDS coordinated change master key can be done by using the ICSF Key Data Set Management panels (see “Changing the master keys” on page 140) or by writing an application to start the Coordinated KDS Administration (CSFCRC) callable service. For the description of the service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*z/OS Cryptographic Services ICSF Application Programmer's Guide.

The Set Master Key utility is used to set the master key on a single LPAR.

Cryptographic features

The cryptographic coprocessors on your systems depends on your configuration. Each coprocessor is capable of performing cryptographic functions and holding master keys within a secure boundary. Table 37 lists the CCA coprocessors, the servers that they are available on, and the supported master keys:

Table 37. Cryptographic coprocessors and master keys

Server	Cryptographic coprocessor	Master keys
z9 EC and z9 BC	CEX2C	AES, DES, and RSA
z10 EC and z10 BC	CEX2C and CEX3C	AES, DES, and RSA
z114 and z196	CEX3C	AES, DES, ECC, and RSA
zEC12 and zBC12	CEX3C and CEX4C	AES, DES, ECC, and RSA
z13 and z13s	CEX5C	AES, DES, ECC, and RSA

Note: The cryptographic accelerators improve clear key RSA operation performance. They do not require the setting of master keys.

Identification of cryptographic features

Starting in ICSF FMID HCR77B0, the prefix used to identify Crypto Express2, Crypto Express3, and Crypto Express4 features has changed. Table 1 on page 5 lists the prefix for these features for FMIDs prior to HCR77B0 and the prefix for these features for FMID HCR77B0 and later releases. This change applies to ICSF messages, panels, and publications. The TKE workstation uses this same identification starting with TKE release 8.0.

Table 38. Cryptographic feature identification

Cryptographic feature	Prefix for FMIDs prior to HCR77B0	Prefix for FMID HCR77B0 and later
Crypto Express2 coprocessor	E	2C
Crypto Express2 accelerator	F	2A
Crypto Express3 coprocessor	G	3C
Crypto Express3 accelerator	H	3A
Crypto Express4 CCA coprocessor	SC	4C
Crypto Express4 EP11 coprocessor	SP	4P
Crypto Express4 accelerator	SA	4A

Note: All newer cryptographic features use the convention where *n* is the number in the feature name. For regional cryptographic servers, *n* represents the generation number of the server:

*n***A**	Crypto Express*n* accelerators.
*n***C**	Crypto Express*n* CCA coprocessors.
*n***P**	Crypto Express*n* EP11 coprocessors.
*n***R**	Regional cryptographic servers. Note: 2R requires ICSF FMID HCR77B1 with PTF OA49069 or later.

New master keys automatically set when ICSF started

ICSF will automatically set new master key registers when ICSF is started if the current master key does not match the master key verification pattern (MKVP) in the key data set. When running with COMPAT(NO), the check of the new master key registers is made every time ICSF is started. When running with COMPAT(YES) or COMPAT(COEXIST), the check is only done when ICSF is started for the first time after an IPL.

The following conditions must be true:

- The new master key register must be full (final key part loaded).
- The MKVP of the new master key register must match the MKVP in the header of the CKDS or PKDS.

Coprocessor activation

In releases of ICSF before FMID HCR7780, in order for a coprocessor to become active, a DES master key is needed to be set on the coprocessor for the coprocessor to become active. Support of any other master key on the coprocessor would then be available.

For FMIDs HCR7780, HCR7790, and HCR77A0, the activation procedure was designed to maximize the number of active coprocessors by selecting the set of master keys that are available on most of the coprocessors. A DES master key is no longer required in order for a coprocessor to become active. Instead, any one of four master keys, the DES master key, the AES master key, the RSA master key (which was called the asymmetric master key in earlier releases), or the ECC master key, is enough for a coprocessor to become active. However, because the goal is to select the combination of master keys that maximize the number of active coprocessors, if a certain master key is not set on all the same coprocessors then that master key support will not be available.

Starting with FMID HCR77A1, the activation procedure uses the master key verification patterns (MKVP) in the header record of the CKDS and PKDS to determine which coprocessors become active. If the MKVP of a master key is in the CKDS or PKDS, that master key must be loaded and the verification pattern of the current master key register must match the MKVP in the CKDS or PKDS. If all of the MKVPs in the CKDS and PKDS match the current master key registers, the coprocessor becomes active. Otherwise, the status of the coprocessor is 'Master key incorrect'.

This applies to all master keys that the coprocessor supports. When there is a MKVP in the CKDS or PKDS and the coprocessor does not support that master key, it is ignored. When a MKVP is not in the CKDS or PKDS, the master key is ignored.

A migration health check is available to find any coprocessors that will not become active when starting ICSF FMID HCR77A1 or later. The ICSFMIG77A1_COPROCESSOR_ACTIVE migration check is available for ICSF FMID HCR7770, HR7780, HCR7790, and HCR77A0.

DES master key

ICSF and the TKE workstation accept a 16-byte key value for the DES master key. The DES master key can be a 128-bit or 192-bit key on the zBC12, zEC12, and later systems with CEX3C or later coprocessor with the September 2012 or later licensed internal code. ICSF and the TKE workstation support loading both key value lengths.

To load a 24-byte DES master key, the **DES master key – 24-byte key** access control point must be enabled in the domain role in all CCA coprocessors for the domain where you wish to use a 24-byte DES master key. If the **DES master key – 24-byte key** access control point is not enabled consistently for all coprocessors for the domain, the DES new master key register cannot be loaded. The master key entry utility will fail. A TKE workstation is required to enable the access control point.

It is not possible to share a CKDS between systems with both 16-byte and 24-byte DES master keys. The master key verification pattern algorithm for the 24-byte DES master key is different from the algorithm for the 16-byte DES master key. The algorithms are described in Appendix C, "Supporting Algorithms and Calculations," on page 397.

The CKDS reencipher and change symmetric master key utilities support both length key values. The coordinated CKDS administration functions support both length key values.

RSA master key

The procedure for loading and changing the RSA master key (RSA-MK) depends on your servers and cryptographic features. The PKA Callable Services control has the same hardware considerations.

If your system is an IBM z9 or IBM z10 server or an IBM z196 or z114 where all CEX3 coprocessors have licensed internal code older than September 2011:

- The PKA callable services control will be active. It will appear on the Administrative Control Functions panel.
- The RSA-MK will be set when the final key part is loaded on the Master Key Entry panel. The master key will be in the current master key register. The PKA callable services control must be disabled to load the RSA-MK
- The PKDS initialization and Update PKDS utilities will use the master key value in the current RSA master key register.
- The RSA-MK can be set using the TKE workstation.
- The coordinated PKDS change master key utility can be used if the RSA-MK is loaded using the TKE workstation. The RSA-MK should not be set using the TKE workstation. The change master key utility will set the master key.

If your system is an IBM z196 or z114 where there is a CEX3 coprocessor with September 2011 or newer licensed internal code or your system is an IBM zEC12, zBC12, or later server:

- The PKA callable services control will not be active. It will not appear on the Administrative Control Functions panel.
- The RSA-MK will not be set when the final key part is loaded on the Master Key Entry panel. The master key will be in the new master key register.
- The PKDS initialization and Update PKDS utilities will use the master key value in the new RSA master key register.

- The TKE workstation cannot be used to set the RSA-MK. Note that new versions of the TKE workstation can be used to set the master keys. This function should not be used as the value of the RSA-MK will not be in the new master key register where ICSF will look for it.
- The RSA-MK can be loaded on a coprocessor (either a new coprocessor or after the master keys are cleared) and set by using the Set MK utility on the Key Data Set Management panel.

PKA Callable Services control

The PKA Callable Services control is used on all z9 and z10 systems and on z196 and z114 systems without a CEX3 coprocessor with the September 2011 or later licensed internal code installed. Newer systems do not use the PKA Callable Services control.

The PKA Callable Services control is used to control the availability of the RSA algorithm. When the control is enabled and the RSA master key is loaded and matches the MKVP in the PKDS, callable services that require the RSA master key will run. When the control is disabled, services that require the RSA master key will fail.

When you enter the RSA master key the first time, the PKA callable services control is initially disabled. Once you have entered the RSA master key and initialized the PKDS, the PKA callable services control will be enabled automatically. When you change the RSA master key, you need to disable the PKA callable services control. To enable and disable the PKA callable services control, see "Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls."

Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls

The dynamic KDS access controls are used to disable services that update key data sets. These controls allow the ICSF administrator to prevent the key data sets from being updated while using utilities that change the key data set. It is recommended that the CKDS and PKDS dynamic services be disabled during the local master key change. This is not necessary when using the coordinated change master key procedure.

When the dynamic CKDS access control is disabled, the callable services that update the DASD copy of the CKDS fails. The affected services are CSNBGIM, CSNBKPI, CSNBKPI2, CSNBKRC, CSNBKRC2, CSNBKRD, CSNBKRW, CSNBKRW2, and CSNBRKA.

When the dynamic PKDS access control is disabled, the callable services that update the DASD copy of the PKDS fails. The affected services are CSNDKRC, CSNDKRD, CSNDKRW, CSNDPKG, CSNDPKI, and CSNDRKD.

When you enter or change the RSA master key, you must first disable the PKA callable services control if it is active. This requirement applies only to the RSA master key. You do not need to disable PKA callable services control in order to enter or change the ECC master key.

Note: The PKA callable services control is only used on z9 and z10 systems and on z196 and z114 systems without a CEX3 coprocessor with the September 2011 or later licensed internal code installed. Newer systems do not use the PKA callable services control. When the control is not active, all steps that refer to the control can be ignored.

1. From the ICSF Primary Menu, select option 4, ADMINCNTL.

```
HCR77C0 ------------- Integrated Cryptographic Service Facility ---------

OPTION ===>

Enter the number of the desired option.

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

Figure 9. Selecting ADMINCNTL on the ICSF primary menu panel

2. The Administrative Control Functions panel appears.

```
 CSFACF00 ------------- ICSF Administrative Control Functions
 COMMAND ===>
          Active CKDS: CSF.CKDS
          Active PKDS: CSF.PKDS
          Active TKDS: CSF.TKDS

To change the status of a control, enter the appropriate character
(E - ENABLE, D - DISABLE) and press ENTER.

         Function                         STATUS
         --------                         ------
 .  Dynamic CKDS Access                   ENABLED
 .  PKA Callable Services                 ENABLED
 .  Dynamic PKDS Access                   ENABLED
```

Figure 10. Selecting ADMINCNTL on the ICSF primary menu panel

   Enter the appropriate character and press ENTER.
   - To enable the dynamic CKDS update services control, enter an 'E' before the Dynamic CKDS Access function.
   - To disable the dynamic CKDS update services control, enter a 'D' before the Dynamic CKDS Access function.
   - To enable the dynamic PKDS update services control, enter an 'E' before the Dynamic PKDS Access function.
   - To disable the dynamic PKDS update services control, enter a 'D' before the Dynamic PKDS Access function.
   - To enable the PKA callable services control, enter an 'E' before the PKA Callable Services function.
   - To disable the PKA callable services control, enter a 'D' before the PKA Callable Services function.

Entering master key parts

You can use the Master Key Entry panels to enter clear master key parts. The way you obtain master key parts depends on the security guidelines in your enterprise. You may receive master key parts from a key distribution center or you may generate your own key parts using the ICSF random number utility.

Important: Regardless of the source of the master key parts or the method of key entry, **make sure the key parts are recorded and saved in a secure location**. When you are entering the key parts for the first time, be aware that **you may need to reenter these same key values at a later date** to restore master key values that have been cleared.

These procedures apply to all master keys supported on your system. The DES master key is used as an example. All of the master keys are optional. You should load the master keys required for your applications.

To enter master key parts that you do not generate using the random number utility, continue with "Steps for entering the first master key part" on page 117.

To begin master key entry by generating random numbers for the key parts, continue with "Generating master key data for master key entry."

Generating master key data for master key entry

If you intend to use the key entry panels to enter master keys, you need to generate and record these values when you begin:

- Key parts.
- Checksums.
- Verification patterns (optional).
- Hash patterns (optional).

Note: If you are reentering master keys when they have been cleared, use the same master key part values as when you originally entered the keys. You should have saved the key part values in a secure place when you entered the master keys previously.

The DES master key (DES-MK) is 16 or 24 bytes long. ICSF defines these master keys by exclusive ORing two or more key parts. Each of the master key parts is also 16 or 24 bytes long. To enter a DES-MK, you must enter a first key part and a final key part. If you choose to, you can also enter one or more intermediate key parts when entering the first key part and the final key part.

Note: The combined DES-MK master key is forced to have odd parity, but the parity of the individual key parts can be odd, even, or mixed. Even or mixed parity keys are referred to as non-odd parity keys.
Attention: The cryptographic coprocessors will not allow certain 'weak' keys as DES and RSA master keys. The list of weak keys are documented in Appendix H, "Questionable (Weak) Keys," on page 421.

The AES master key (AES-MK) is 32 bytes long. ICSF defines these master keys by exclusive ORing two or more key parts.

The RSA master key (RSA-MK) is 24 bytes long. ICSF defines these master keys by exclusive ORing two or more key parts.

The ECC master key (ECC-MK) is 32-bytes long. ICSF defines these master keys by exclusive ORing two or more key parts.

If you are using ICSF to generate random numbers, generate a random number for each key part that you need to enter to create the master key.

A 16-byte key part consists of 32 hexadecimal digits. A 24-byte key part consists of 48 hexadecimal digits. To make this process easier, each part is broken into segments of 16 digits each. A 32-byte key part consists of 64 hexadecimal digits.

When you are manually entering the master key parts, you also enter a checksum that verifies whether you entered the key part correctly. A checksum is a two-digit result of putting a key part value through a series of calculations. The coprocessors calculate the checksum with the key part you enter and compare the one they calculated with the one you entered. The checksum verifies that you did not transpose any digits when entering the key part. If the checksums are equal, you have successfully entered the key.

When you enter a key part and its checksum for a DES-MK, the coprocessor calculates an eight-byte verification pattern and sixteen-byte hash pattern. When you enter a key part and its checksum for a AES-MK, the coprocessor calculates an eight-byte verification pattern. When you enter a key part and its checksum for the RSA-MK, the coprocessor calculates a sixteen-byte verification pattern. When you enter a key part and its checksum for an ECC-MK, the coprocessor calculates an eight-byte verification pattern.

The ICSF Master Key Entry panel displays the verification pattern. Check the displayed verification pattern against the option verification pattern you may have generated at the time you generated the AES, DES, ECC, or RSA master key parts. The verification pattern checks whether you entered the key part correctly and whether you entered the correct key type.

ICSF displays a verification and/or hash pattern for each master key part. It also displays a verification and/or hash pattern for the master key when you enter all the key parts. If the verification and hash patterns are the same, you have entered the key parts correctly.

To generate the value for a key part, you can use one of these methods:
- Choose a random number yourself.
- Access the ICSF utility panels to generate a random number.
- Call the random number generate callable service. For more information, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Steps for generating key parts using ICSF utilities

1. From the ICSF Primary Menu, select option 5, UTILITY.

```
HCR77C0 ------------- Integrated Cryptographic Service Facility ---------

OPTION ===>

Enter the number of the desired option.

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

Figure 11. Selecting UTILITY on the ICSF primary menu panel

2. The Utilities panel appears. You use the RANDOM and CHECKSUM options to generate random numbers, checksums, and verification patterns for master key management. Select option 3, RANDOM, to access the Random Number Generator panel.

```
CSFUTL00 ---------------- ICSF - Utilities -------------------------
OPTION ===>


Enter the number of the desired option above.

  1  ENCODE        -  Encode data
  2  DECODE        -  Decode data
  3  RANDOM        -  Generate a random number
  4  CHECKSUM      -  Generate a checksum and verification and
                      hash patterns
  5  PPKEYS        -  Generate master key values from a pass phrase
  6  PKDSKEYS      -  Manage keys in the PKDS
```

Figure 12. Selecting RANDOM on the ICSF Utilities panel

3. The Random Number Generator panel appears. To select the parity of the random numbers, enter ODD, EVEN, or RANDOM next to Parity Option and press ENTER.

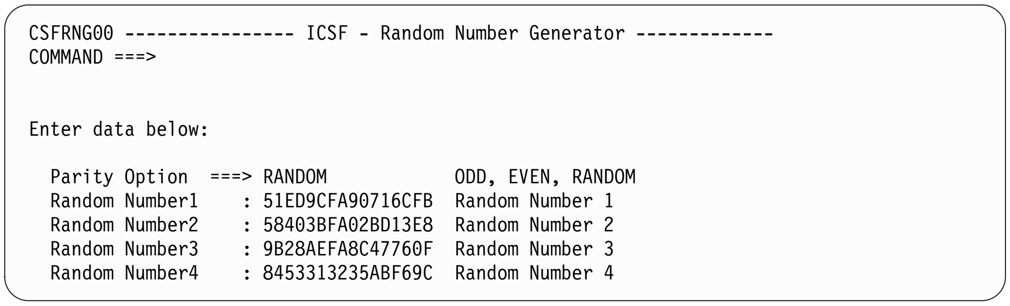
```
CSFRNG00 ---------------- ICSF - Random Number Generator -------------
COMMAND ===>


Enter data below:

  Parity Option  ===>                  ODD, EVEN, RANDOM
  Random Number1    : 0000000000000000  Random Number 1
  Random Number2    : 0000000000000000  Random Number 2
  Random Number3    : 0000000000000000  Random Number 3
  Random Number4    : 0000000000000000  Random Number 4
```

Figure 13. Selecting Parity on the Random Number Generator panel

The DES master key is forced to have odd parity, regardless of the parity option you select for each key part. Parity is not checked for AES, ECC, or RSA master keys.

A random 16-digit number appears in each of the Random Number fields. You can use each of these random numbers for a segment of a key part.

The DES master key uses random numbers 1 and 2 or 1 through 3 if you are using a 24-byte master key. The RSA master key uses random numbers 1 through 3. The AES and ECC master keys use random numbers 1 through 4.



```
CSFRNG00 ---------------- ICSF - Random Number Generator -------------
COMMAND ===>


Enter data below:

  Parity Option  ===> RANDOM            ODD, EVEN, RANDOM
  Random Number1    : 51ED9CFA90716CFB  Random Number 1
  Random Number2    : 58403BFA02BD13E8  Random Number 2
  Random Number3    : 9B28AEFA8C47760F  Random Number 3
  Random Number4    : 8453313235ABF69C  Random Number 4
```

Figure 14. ICSF Random Number Generator Panel with Generated Numbers

4. When you end the utility panels and access the Master Key Part Entry panel, the key parts you generated are transferred automatically to the Master Key Part Entry panels. For this reason, you will not need to enter the key parts on the Master Key Part Entry panels.

   Although the key parts are automatically transferred to the Master Key Entry panels, make sure you **record the random numbers and store them in a safe place**. You must have these numbers in case you ever need to reenter the master key values. If you ever need to restore a master key that has been cleared for any reason, you will need the key part values.
5. Press END to return to the Utilities panel.
6. Continue with "Steps for generating a checksum, verification pattern, or hash pattern for a key part."

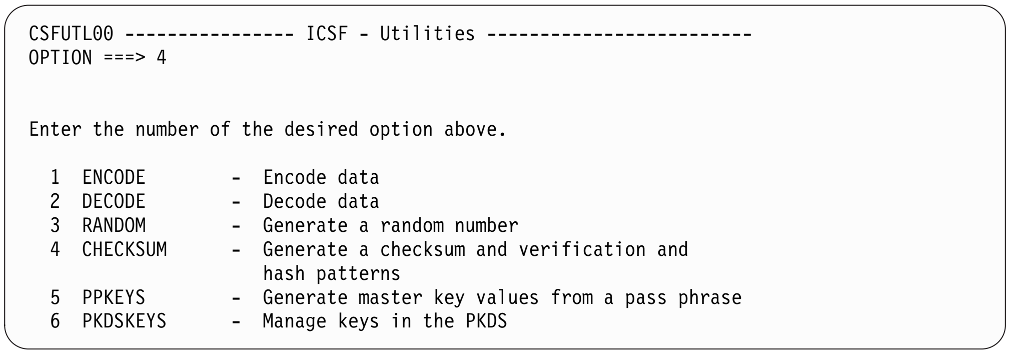
Steps for generating a checksum, verification pattern, or hash pattern for a key part

You can use the Utilities panel to generate a checksum and either an optional verification pattern or an optional hash pattern for a key part. You can use this panel to generate a checksum for a key part even if ICSF has not been initialized.

Note: The use of the Utilities panel to generate the key part, the checksum, and the verification pattern exposes the key part in storage for the duration of the dialogs. For this reason, you can choose to calculate both the checksum, the verification pattern, or the hash pattern values manually or by using a PC program. See "Checksum Algorithm" on page 397 for a description of the checksum algorithm. See "Algorithm for calculating a verification pattern" on page 399 for a description of the algorithm for the verification pattern. See "The MDC–4 Algorithm for Generating Hash Patterns" on page 400 for a description of the MDC-4 algorithm that is used to calculate a hash pattern for a key part. The use of the verification pattern or hash pattern is optional.

Follow these steps to generate the checksum and the optional verification pattern or hash pattern for a key part.

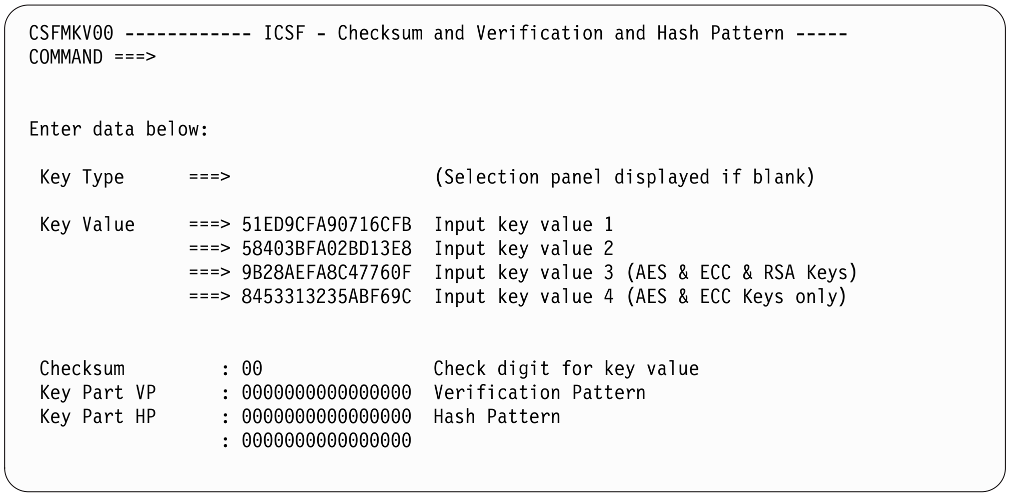
1. Select option 4, CHECKSUM, on the ICSF Utilities panel as shown in Figure 15 on page 115.



```
CSFUTL00 ---------------- ICSF - Utilities -------------------------
OPTION ===> 4


Enter the number of the desired option above.

  1  ENCODE        -  Encode data
  2  DECODE        -  Decode data
  3  RANDOM        -  Generate a random number
  4  CHECKSUM      -  Generate a checksum and verification and
                      hash patterns
  5  PPKEYS        -  Generate master key values from a pass phrase
  6  PKDSKEYS      -  Manage keys in the PKDS
```

Figure 15. Selecting the Checksum Option on the ICSF Utilities Panel

The Checksum and Verification and Hash Pattern panel appears. See Figure 16.



```
CSFMKV00 ------------ ICSF - Checksum and Verification and Hash Pattern -----
COMMAND ===>


Enter data below:

 Key Type      ===>                   (Selection panel displayed if blank)

 Key Value     ===> 51ED9CFA90716CFB  Input key value 1
               ===> 58403BFA02BD13E8  Input key value 2
               ===> 9B28AEFA8C47760F  Input key value 3 (AES & ECC & RSA Keys)
               ===> 8453313235ABF69C  Input key value 4 (AES & ECC Keys only)


 Checksum         : 00                Check digit for key value
 Key Part VP      : 0000000000000000  Verification Pattern
 Key Part HP      : 0000000000000000  Hash Pattern
                  : 0000000000000000
```

Figure 16. ICSF Checksum and Verification and Hash Pattern Panel

If you accessed the Random Number Generator panel prior to this panel, the random numbers that are generated appear automatically in the Key Value fields.

2. If you did not use the Random Number Generator panel to generate random numbers, enter the numbers for which you want to create checksum, verification pattern, or hash patterns into the key value fields. Because these will be the key part values you will specify in the Master Key Entry panels, make sure you record the numbers.
3. In the Key Type field, specify either:
   - DES-MK to generate a checksum, hash, and verification pattern for a 16-byte DES master key part.
   - DES24-MK to generate a checksum, hash, and verification pattern for a 24-byte DES master key part.
   - AES-MK to generate a checksum and verification pattern for an AES master key part.
   - RSA-MK to generate a checksum and verification pattern for an RSA master key part.

- ECC-MK to generate a checksum and verification pattern for an ECC master key part.

  If you leave the Key Type field blank and press ENTER, the Key Type Selection panel appears. See Figure 17.

```
CSFMKV10 ------------- ICSF - Key Type Selection Panel ---- ROW 1 to 9 OF 9
COMMAND ===>                                                SCROLL ===> PAGE

Select one key type only
    KEY TYPE      DESCRIPTION
    AES-MK        AES Master Key
    ASYM-MK       Asymmetric Master key
    DES-MK        DES Master key (16-byte)
    DES24-MK      DES Master key (24-byte)
    ECC-MK        ECC Master key
    EXPORTER      Export key encrypting key
    IMP-PKA       Limited Authority Importer key
    IMPORTER      Import key encrypting key
    IPINENC       Input PIN encrypting key
    OPINENC       Output PIN encrypting key
    PINGEN        PIN generation key
    PINVER        PIN verification key
    RSA-MK        RSA Master key
***************************** BOTTOM OF DATA *****************************
```

Figure 17. Key Type Selection Panel Displayed During Hardware Key Entry

4. Type 'S' to the left of the DES-MK key type, and press ENTER to return to the Checksum and Verification Pattern panel as shown in Figure 18.

   In this example, we have selected the DES-MK master key.

```
CSFMKV00 ------ ICSF - Checksum and Verification and Hash Pattern ---
 COMMAND ===>


 Enter data below:

Key Type      ===> DES-MK            (Selection panel displayed if blank)

   Key Value     ===> 51ED9CFA90716CFB  Input key value 1
                 ===> 58403BFA02BD13E8  Input key value 2
                 ===> 0000000000000000  Input key value 3 (AES & ECC & RSA Keys)
                 ===> 0000000000000000  Input key value 4 (AES & ECC Keys only)

   Checksum         : 40                   Check digit for key part
   Key Part VP      : 0CCE190A635A6C89  Verification Pattern
   Key Part HP      : EA58E51179754FB7  Hash Pattern
                    : C102957465CE479E
```

Figure 18. ICSF Checksum and Verification Pattern Panel

5. *Record the checksum, verification pattern, and hash pattern.*

   Save these values in a secure place along with the key part values in case you need to reenter the key values. If the cryptographic feature detects tampering, it clears the master key, and you have to reenter the same master key again.
6. Press END to return to the Utilities panel.
7. Press END again to return to the ICSF Primary menu.

Continue with the appropriate topic for steps to enter the master key part you have just generated.

- If you have generated the first master key part, continue with “Steps for entering the first master key part.”
- If you have generated an intermediate master key part, continue with “Steps for entering intermediate key parts” on page 120.
- If you have generated a final master key part, continue with “Steps for entering the final key part” on page 121.

Steps for entering the first master key part

Use the Master Key Entry panels to enter each key part. You can enter as many key parts as you like. When the new master key register is empty, the first key part must be identified as FIRST. Subsequent intermediate key parts must be identified as MIDDLE. To close the new master key register to prevent additional key parts from being loaded, the final key part must be identified as FINAL.

Important: When entering the key part values, be aware that **you may need to reenter these same key values at a later date** to restore master key values that have been cleared. Make sure the key part values are recorded and saved in a secure location.

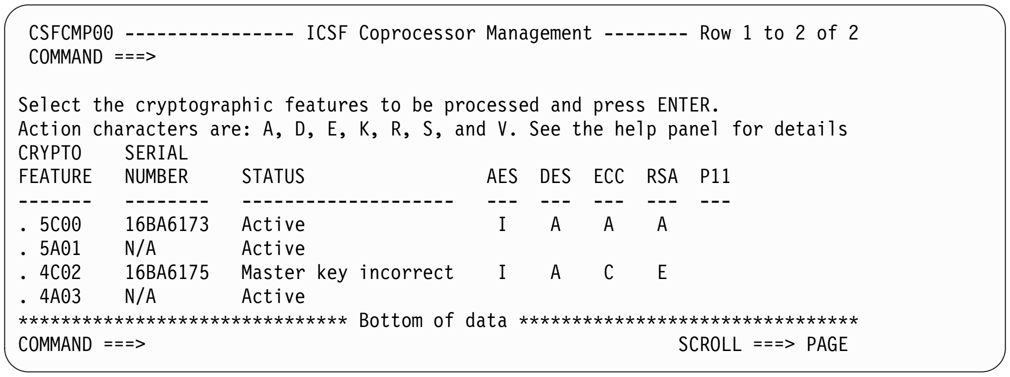
If you use the random number generator utility to generate key parts, enter each key part directly after you generate the key part data and prior to generating another key part.

To enter master key parts:

1. Select option 1, COPROCESSOR MGMT, on the “ICSF Primary Menu panel” on page 387, and press ENTER.

   The ICSF Coprocessor Management panel appears (Figure 19).
2. Select the coprocessor or coprocessors to be processed by entering an 'E' and then pressing ENTER. Select as many coprocessors as required. This loads the same master key for all coprocessors selected.

   Note: During first time initialization, the coprocessor status will be 'Master key incorrect'. When master keys have been set, the status will be 'Active'.



```
 CSFCMP00 ---------------- ICSF Coprocessor Management -------- Row 1 to 2 of 2
 COMMAND ===>

Select the cryptographic features to be processed and press ENTER.
Action characters are: A, D, E, K, R, S, and V. See the help panel for details
CRYPTO    SERIAL
FEATURE   NUMBER     STATUS                 AES  DES  ECC  RSA  P11
-------   --------   --------------------   ---  ---  ---  ---  ---
. 5C00    16BA6173   Active                  I    A    A    A
. 5A01    N/A        Active
. 4C02    16BA6175   Master key incorrect    I    A    C    E
. 4A03    N/A        Active
******************************* Bottom of data ********************************
COMMAND ===>                                              SCROLL ===> PAGE
```

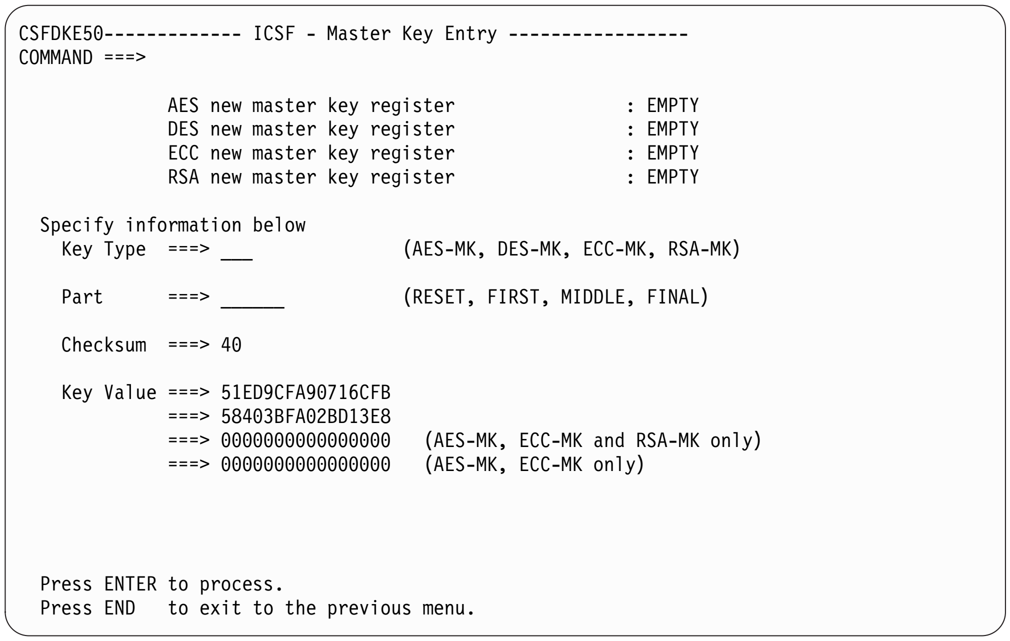
Figure 19. Selecting the coprocessor on the Coprocessor Management Panel

The coprocessor management panels shows all accelerators and coprocessors, their status, and the state of the master keys for coprocessors. Accelerators do not have master keys and the states are blank. When a coprocessor does not support a master key, a hyphen (-) is used for its state. The master key state for coprocessors shows U (uninitialized), C (correct), A (active), E (error), and I (ignored).

Coprocessor activation uses the master key verification patterns (MKVP) in the header record of the CKDS and PKDS to determine which coprocessors become active. If the MKVP of a master key is in the CKDS or PKDS, that master key must be loaded and the verification pattern of the current master key register must match the MKVP in the CKDS or PKDS. If all of the MKVPs in the CKDS and PKDS match the current master key registers, the coprocessor will become active. Otherwise, the status of the coprocessor is 'Master key incorrect'.

This applies to all master keys that the coprocessor supports. When there is a MKVP in the CKDS or PKDS and the coprocessor does not support that master key, it is ignored. When a MKVP is not in the CKDS or PKDS, the master key is ignored.

3. The ICSF Master Key Entry panel appears. See Figure 20.



```
CSFDKE50------------- ICSF - Master Key Entry -----------------
COMMAND ===>

              AES new master key register                : EMPTY
              DES new master key register                : EMPTY
              ECC new master key register                : EMPTY
              RSA new master key register                : EMPTY

  Specify information below
    Key Type  ===> ___             (AES-MK, DES-MK, ECC-MK, RSA-MK)

    Part      ===> ______          (RESET, FIRST, MIDDLE, FINAL)

    Checksum  ===> 40

    Key Value ===> 51ED9CFA90716CFB
              ===> 58403BFA02BD13E8
              ===> 0000000000000000   (AES-MK, ECC-MK and RSA-MK only)
              ===> 0000000000000000   (AES-MK, ECC-MK only)



  Press ENTER to process.
  Press END   to exit to the previous menu.
```

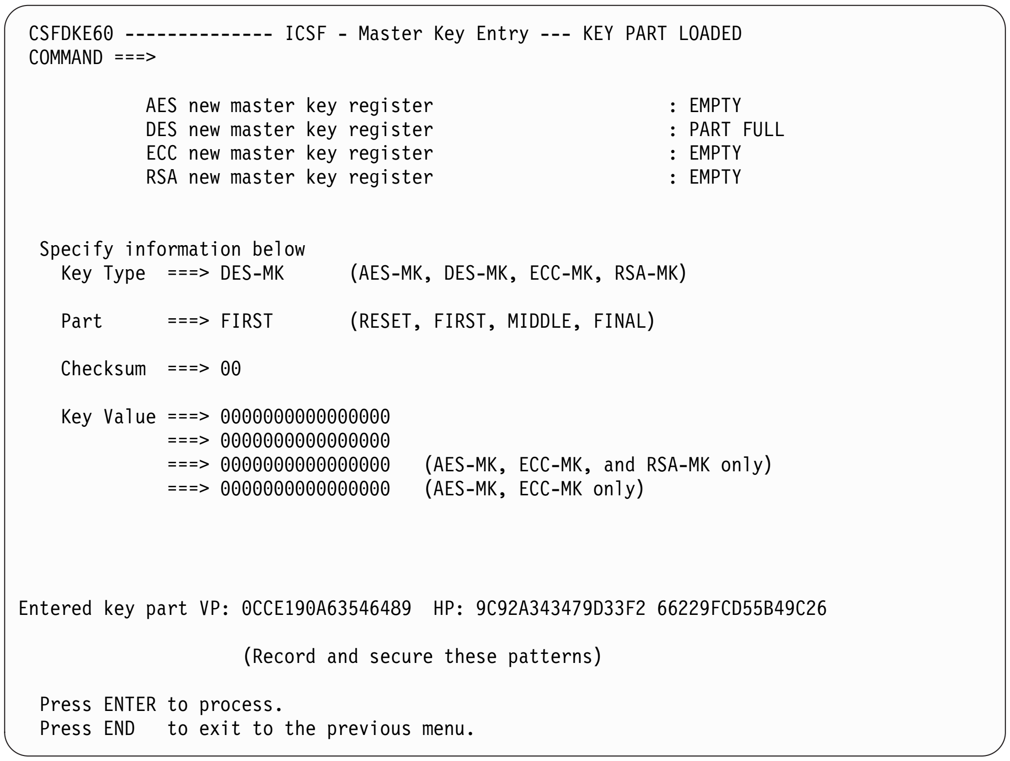
Figure 20. Master Key Entry Panel

4. Fill in the panel
   a. Enter the master key type in the Key Type field.

      In this example we are entering the DES-MK master key.
   b. Enter FIRST in the Part field.
   c. Enter the two-digit checksum and the two 16-digit key values if you did not use the random number panel.
   d. Make sure you have recorded the two 16-digit key values. You may need to reenter these same values at a later date to restore master key values that have been cleared. **Make sure all master key parts you enter are recorded and saved in a secure location.**
   e. When all the fields are complete, press ENTER.

If the checksum entered in the checksum field matches the checksum that the master key entry utility calculated, the key part is accepted. The message at the top of the panel states KEY PART LOADED, as shown in Figure 21. The new master key register status changes to PART FULL. The verification pattern and hash pattern that are calculated for the key part appear near the bottom of the panel. Compare them with the patterns generated by the random number generator or provided by the person who gave you the key part value to enter.

f. Record the verification pattern and hash pattern.



```
 CSFDKE60 ------------- ICSF - Master Key Entry --- KEY PART LOADED
 COMMAND ===>

            AES new master key register                      : EMPTY
            DES new master key register                      : PART FULL
            ECC new master key register                      : EMPTY
            RSA new master key register                      : EMPTY


  Specify information below
    Key Type  ===> DES-MK      (AES-MK, DES-MK, ECC-MK, RSA-MK)

    Part      ===> FIRST       (RESET, FIRST, MIDDLE, FINAL)

    Checksum  ===> 00

    Key Value ===> 0000000000000000
              ===> 0000000000000000
              ===> 0000000000000000   (AES-MK, ECC-MK, and RSA-MK only)
              ===> 0000000000000000   (AES-MK, ECC-MK only)



Entered key part VP: 0CCE190A63546489  HP: 9C92A343479D33F2 66229FCD55B49C26

                     (Record and secure these patterns)

  Press ENTER to process.
  Press END   to exit to the previous menu.
```

Figure 21. The Master Key Entry Panel Following Key Part Entry

5. If the checksums do not match, the message Invalid Checksum appears. If this occurs, follow this sequence to resolve the problem:
   a. Reenter the checksum.
   b. If you still get a checksum error, recalculate the checksum.
   c. If your calculations result in a different value for the checksum, enter the new value.
   d. If your calculations result in the same value for the checksum, or if a new checksum value does not resolve the error, reenter the key part halves and checksum.

When you have entered the first key part successfully, continue with:

- “Steps for generating key parts using ICSF utilities” on page 112 if you are using the ICSF utilities to generate random numbers for key values.
- “Steps for entering intermediate key parts” on page 120 if you are entering key parts manually.

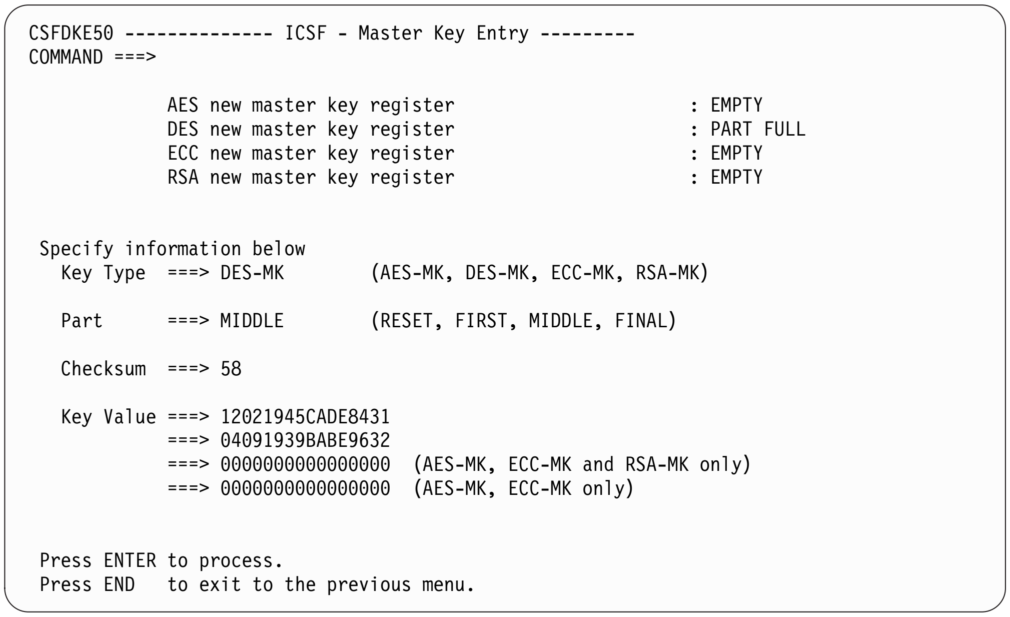
Steps for entering intermediate key parts

If you want to enter more than two key parts, you must enter one or more intermediate key parts. Enter intermediate key parts after you enter the first key part and prior to entering the final one.

To enter intermediate master key parts:

1. Select option 1, COPROCESSOR MGMT, on the ICSF Primary menu and press ENTER.

   The Coprocessor Management panel appears.
2. Select the coprocessor or coprocessors to be processed by entering an 'E' on the Coprocessor Management panel. Select the same coprocessors that were selected when entering the first key value.
3. When pressing ENTER, the Master Key Entry panel appears (Figure 22).



```
CSFDKE50 -------------- ICSF - Master Key Entry ---------
COMMAND ===>

             AES new master key register                     : EMPTY
             DES new master key register                     : PART FULL
             ECC new master key register                     : EMPTY
             RSA new master key register                     : EMPTY


 Specify information below
   Key Type  ===> DES-MK        (AES-MK, DES-MK, ECC-MK, RSA-MK)

   Part      ===> MIDDLE        (RESET, FIRST, MIDDLE, FINAL)

   Checksum  ===> 58

   Key Value ===> 12021945CADE8431
             ===> 04091939BABE9632
             ===> 0000000000000000  (AES-MK, ECC-MK and RSA-MK only)
             ===> 0000000000000000  (AES-MK, ECC-MK only)


 Press ENTER to process.
 Press END   to exit to the previous menu.
```

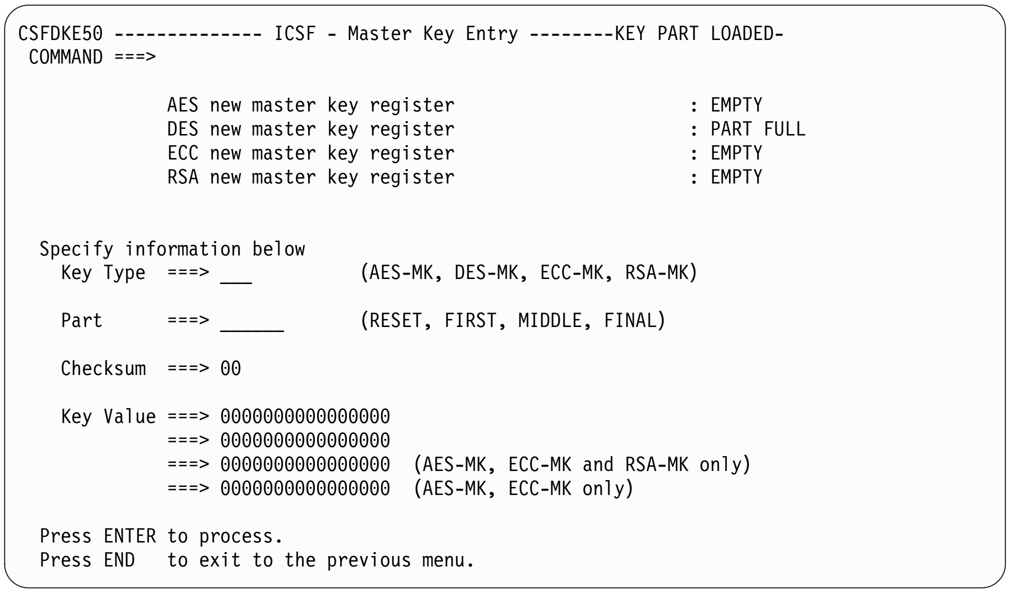
Figure 22. The Master Key Entry Panel for Intermediate Key Values

4. Fill in the panel
   a. Enter the master key type in the Key Type field. In this example we are continuing to enter the DES master key.
   b. Enter MIDDLE in the Part field.
   c. Enter the two-digit checksum and the two 16-digit key values if you did not use random number panel.
   d. Make sure you have recorded the two 16-digit key values. You may need to reenter these same values at a later date to restore master key values that have been cleared. **Make sure all master key parts you enter are recorded and saved in a secure location.**
   e. When all the fields are complete, press ENTER.

      If the checksum entered in the checksum field matches the checksum that the master key entry utility calculated, the key part is accepted. The message at the top of the panel states `KEY PART LOADED`, as shown in Figure 23 on page 121. The new master key register status changes to `PART`

FULL. The verification pattern and hash pattern that are calculated for the key part appear near the bottom of the panel. Compare them with the patterns generated by the random number generator or provided by the person who gave you the key part value to enter.

f. Record the verification pattern and hash pattern.



```
CSFDKE50 -------------- ICSF - Master Key Entry --------KEY PART LOADED-
 COMMAND ===>

              AES new master key register                      : EMPTY
              DES new master key register                      : PART FULL
              ECC new master key register                      : EMPTY
              RSA new master key register                      : EMPTY


  Specify information below
    Key Type  ===> ___         (AES-MK, DES-MK, ECC-MK, RSA-MK)

    Part      ===> ______      (RESET, FIRST, MIDDLE, FINAL)

    Checksum  ===> 00

    Key Value ===> 0000000000000000
              ===> 0000000000000000
              ===> 0000000000000000  (AES-MK, ECC-MK and RSA-MK only)
              ===> 0000000000000000  (AES-MK, ECC-MK only)

  Press ENTER to process.
  Press END   to exit to the previous menu.
```

Figure 23. The Master Key Entry Panel with Intermediate Key Values

5. If the checksums do not match, the message Invalid Checksum appears. If this occurs, follow this sequence to resolve the problem:
   a. Reenter the checksum.
   b. If you still get a checksum error, recalculate the checksum.
   c. If your calculations result in a different value for the checksum, enter the new value.
   d. If your calculations result in the same value for the checksum or if a new checksum value does not resolve the error, reenter the key part halves and checksum.

When you have entered the middle key part successfully, continue with:
- "Steps for generating key parts using ICSF utilities" on page 112 if you are using the ICSF utilities to generate random numbers for key values.
- "Steps for entering the final key part" if you are entering key parts manually.

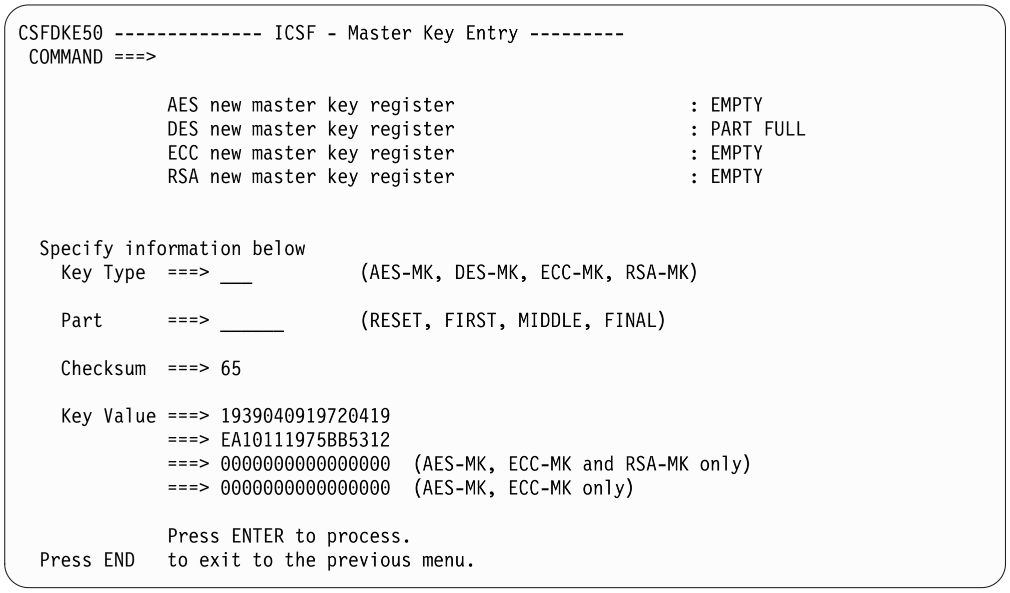
Steps for entering the final key part

After you enter the first key part and any intermediate key parts, you then enter the final master key part.

1. Select option 1, COPROCESSOR MGMT, on the ICSF Primary menu and press ENTER.

   The Coprocessor Management panel appears.
2. Select the coprocessor or coprocessors to be processed by entering an 'E' on the Coprocessor Management panel.

3. When pressing ENTER, the Master Key Entry panel appears.



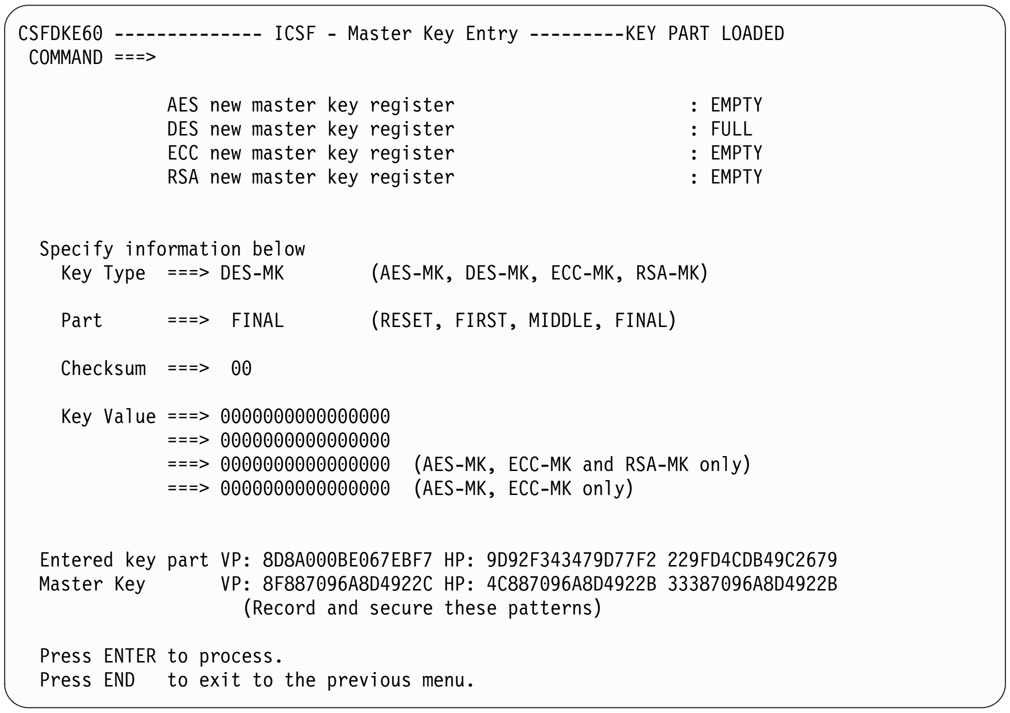
```
CSFDKE50 -------------- ICSF - Master Key Entry ---------
 COMMAND ===>

              AES new master key register                      : EMPTY
              DES new master key register                      : PART FULL
              ECC new master key register                      : EMPTY
              RSA new master key register                      : EMPTY


  Specify information below
    Key Type  ===> ___          (AES-MK, DES-MK, ECC-MK, RSA-MK)

    Part      ===> ______       (RESET, FIRST, MIDDLE, FINAL)

    Checksum  ===> 65

    Key Value ===> 1939040919720419
              ===> EA10111975BB5312
              ===> 0000000000000000  (AES-MK, ECC-MK and RSA-MK only)
              ===> 0000000000000000  (AES-MK, ECC-MK only)

              Press ENTER to process.
  Press END   to exit to the previous menu.
```

Figure 24. The Master Key Entry Panel when entering Final Key Values

4. Fill in the panel
   a. Enter the master key type in the Key Type field.

      In this example, we are continuing to enter the DES master key.
   b. Enter FINAL in the Part field.
   c. Enter the two-digit checksum and the two 16-digit key values if you did not use random number panel.
   d. Make sure you have recorded the two 16-digit key values. You may need to reenter these same values at a later date to restore master key values that have been cleared. **Make sure all master key parts you enter are recorded and saved in a secure location.**
   e. When all the fields are complete, press ENTER.

      If the checksum entered in the checksum field matches the checksum that the master key entry utility calculated, the key part is accepted. The message at the top of the panel states `KEY PART LOADED`, as shown in Figure 25 on page 123. The new master key register status changes to `FULL`. The verification pattern and hash pattern that are calculated for the key part appear near the bottom of the panel. Compare them with the patterns generated by the random number generator or provided by the person who gave you the key part value to enter.
   f. Record the verification pattern and hash pattern.



```
CSFDKE60 -------------- ICSF - Master Key Entry ---------KEY PART LOADED
 COMMAND ===>

             AES new master key register                      : EMPTY
             DES new master key register                      : FULL
             ECC new master key register                      : EMPTY
             RSA new master key register                      : EMPTY


  Specify information below
    Key Type  ===> DES-MK        (AES-MK, DES-MK, ECC-MK, RSA-MK)

    Part      ===>  FINAL        (RESET, FIRST, MIDDLE, FINAL)

    Checksum  ===>  00

    Key Value ===> 0000000000000000
              ===> 0000000000000000
              ===> 0000000000000000  (AES-MK, ECC-MK and RSA-MK only)
              ===> 0000000000000000  (AES-MK, ECC-MK only)


  Entered key part VP: 8D8A000BE067EBF7 HP: 9D92F343479D77F2 229FD4CDB49C2679
  Master Key       VP: 8F887096A8D4922C HP: 4C887096A8D4922B 33387096A8D4922B
                     (Record and secure these patterns)

  Press ENTER to process.
  Press END   to exit to the previous menu.
```

Figure 25. The Master Key Entry Panel with Final Key Values

5. If the checksums do not match, the message `Invalid Checksum` appears. If this occurs, follow this sequence to resolve the problem:
   a. Reenter the checksum.
   b. If you still get a checksum error, recalculate the checksum.
   c. If your calculations result in a different value for the checksum, enter the new value.
   d. If your calculations result in the same value for the checksum or if a new checksum value does not resolve the error, reenter the key part halves and checksum.
6. When you have entered the final key part successfully, it is combined with the first key part and any intermediate key parts in the new master key register.

   The new master key register status is now FULL, and the panel displays two verification patterns and two hash patterns. It gives you verification patterns and hash patterns for both the final key part and the new master key because it is now complete.
7. Check that the key part verification pattern or hash pattern you may have previously calculated matches the verification pattern or hash pattern that is shown on the panel. If they do not, you may want to restart the key entry process. For information on how to restart the key entry process, see "Steps for restarting the key entry process" on page 124.
8. *Record the verification pattern and hash pattern* for the new master key because you may want to verify it at another time.

Note: When you initialize or reencipher a CKDS or PKDS, ICSF places the verification pattern for the master keys into the key data set header record.

When you have entered the master keys correctly, they are in the new master key registers and are not active on the system.

Note: Ensure that the new master key is installed on all cryptographic coprocessors.

When you enter the master keys, you should do *one* of these:

- If you are defining the DES or AES master keys for the first time, initialize the CKDS with the DES and AES master keys. For a description of the process of initializing a CKDS on your system, see "Initializing the key data sets at first-time startup" on page 128.
- If you are defining the ECC or RSA master keys for the first time, initialize the PKDS with the master keys. For a description of the process of initializing a PKDS on your system, see "Initializing the key data sets at first-time startup" on page 128.
- If you are defining an AES, DES, ECC, or RSA master key when it was cleared, set the master keys to make them active. For a description of the process of recovering from tampering, see "Reentering master keys when they have been cleared" on page 126.
- If you are defining the DES or AES master keys to add to an existing CKDS, update the CKDS with the new master key. For a description of the process of updating a CKDS on your system, see "Updating the key data sets with additional master keys" on page 132 for additional information.
- If you are defining the ECC or RSA master keys to add to an existing PKDS, update the PKDS with the new master key. For a description of the process of updating a PKDS on your system, see "Updating the key data sets with additional master keys" on page 132 for additional information.
- If you are changing a DES or AES master key, reencipher the CKDS under the new DES or AES master key and make it active. For a description of the process of changing a DES or AES master key, see "Changing the master keys" on page 140.
- If you are changing an ECC or RSA master keys, reencipher the PKDS under the new ECC or RSA master key and make it active. For a description of the process of changing a ECC or RSA master key, see "Changing the master keys" on page 140.

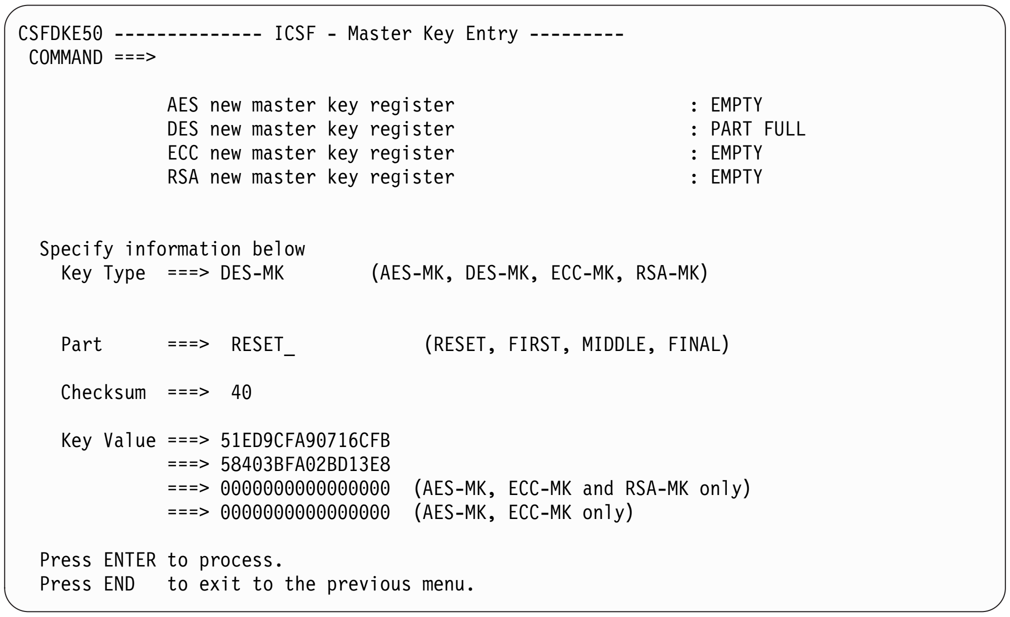
Steps for restarting the key entry process

If you realize that you made an error when entering a key part, you can restart the process of entering the new master key. For example, if the verification pattern or the hash pattern that was calculated does not match the one that you calculated, you may want to restart the process. Restarting the key entry process clears the new master key register, which erases all the new master key parts you entered previously.

To restart the key entry process, follow these steps:

1. On the Master Key Entry panel, enter the master key type in the Key Type field.

   In this example, we are resetting a new DES master key.
2. Enter RESET in the Part field.

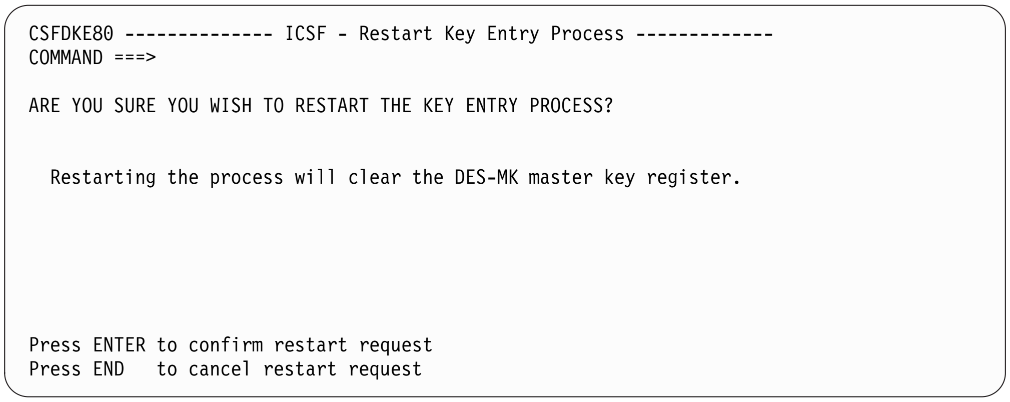


```
CSFDKE50 -------------- ICSF - Master Key Entry ---------
 COMMAND ===>

             AES new master key register                      : EMPTY
             DES new master key register                      : PART FULL
             ECC new master key register                      : EMPTY
             RSA new master key register                      : EMPTY


  Specify information below
    Key Type  ===> DES-MK        (AES-MK, DES-MK, ECC-MK, RSA-MK)


    Part      ===>  RESET_           (RESET, FIRST, MIDDLE, FINAL)

    Checksum  ===>  40

    Key Value ===> 51ED9CFA90716CFB
              ===> 58403BFA02BD13E8
              ===> 0000000000000000  (AES-MK, ECC-MK and RSA-MK only)
              ===> 0000000000000000  (AES-MK, ECC-MK only)

  Press ENTER to process.
  Press END   to exit to the previous menu.
```

Figure 26. Selecting Reset on the Master Key Entry Panel

3. Press ENTER.

   The Restart Key Entry Process panel appears. See Figure 27. This panel confirms your request to restart the key entry process.



```
CSFDKE80 -------------- ICSF - Restart Key Entry Process -------------
COMMAND ===>

ARE YOU SURE YOU WISH TO RESTART THE KEY ENTRY PROCESS?


  Restarting the process will clear the DES-MK master key register.







Press ENTER to confirm restart request
Press END   to cancel restart request
```

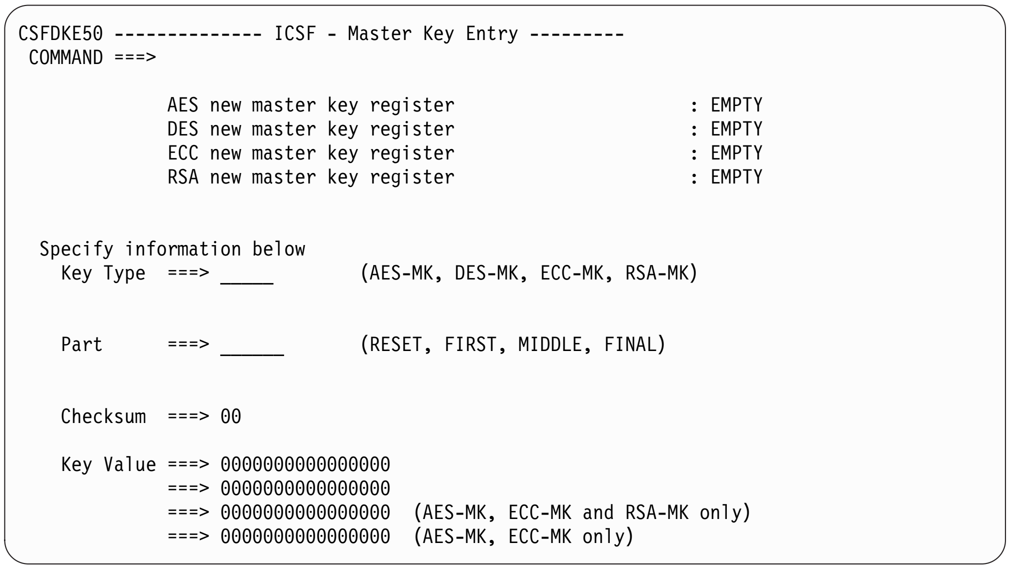
Figure 27. Confirm Restart Request Panel

4. If you want to restart the key entry process, press ENTER.

   The restart request automatically empties the master key register.

5. If you do not want to restart, press END.

   When you make a choice, you return to the Master Key Entry panel. If you selected to continue with the restart process, the new master key register status field is reset to EMPTY, as shown in Figure 28 on page 126. This indicates that the register has been cleared.



```
CSFDKE50 -------------- ICSF - Master Key Entry ---------
 COMMAND ===>

              AES new master key register                      : EMPTY
              DES new master key register                      : EMPTY
              ECC new master key register                      : EMPTY
              RSA new master key register                      : EMPTY


  Specify information below
    Key Type  ===> _____        (AES-MK, DES-MK, ECC-MK, RSA-MK)


    Part      ===> ______       (RESET, FIRST, MIDDLE, FINAL)


    Checksum  ===> 00

    Key Value ===> 0000000000000000
              ===> 0000000000000000
              ===> 0000000000000000  (AES-MK, ECC-MK and RSA-MK only)
              ===> 0000000000000000  (AES-MK, ECC-MK only)
```

Figure 28. The Master Key Entry Panel Following Reset Request

6. Either begin the key entry process again or press END to return to the ICSF primary menu panel.

Reentering master keys when they have been cleared

In these situations, the cryptographic feature clears the master key registers so that the master key values are not disclosed.

- If the cryptographic feature detects tampering (the intrusion latch is tripped), ALL installation data is cleared: master keys, retained keys for all domains, as well as roles and profiles.
- If the cryptographic feature detects tampering (the secure boundary of the card is compromised), the card is rendered inoperable.
- If you issue a command from the TKE workstation to zeroize a domain. This command zeroizes the master key data specific to the domain.
- If you issue a command from the Support Element panels to zeroize all domains. This command zeroizes ALL installation data: master keys, retained keys, and access control roles and profiles.

Although the values of the master keys are cleared, the secure keys in the CKDS and PKDS are still enciphered under the cleared master keys. Therefore, to recover the keys in the CKDS and PKDS, you must reenter the same master keys and set the master key. For security reasons, you may then want to change all the master keys.

PR/SM considerations

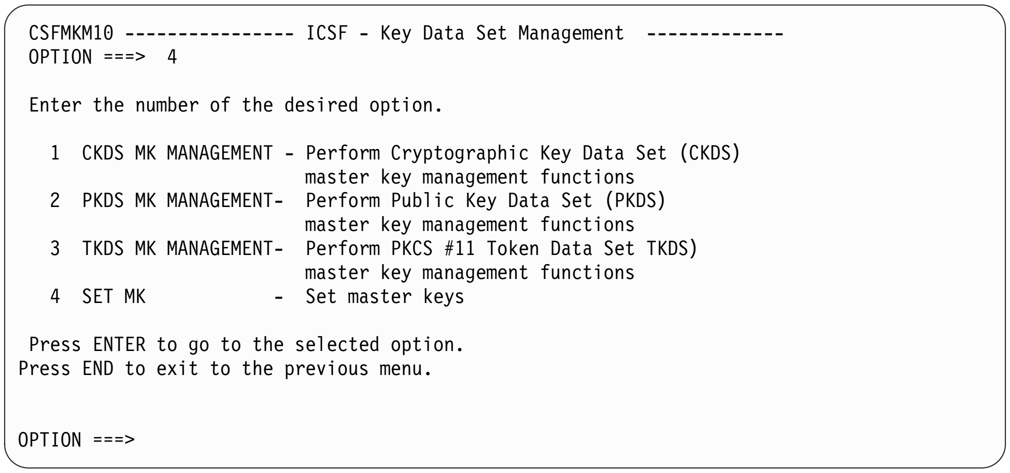
You must first ensure that key entry is enabled for each LP on the Change LPAR Cryptographic Controls panel of the SE. You must then reenter the master keys in each LPAR. If you zeroize a domain using the TKE workstation however, the master keys are cleared only in that domain. Master keys in other domains are not affected and do not need to be reentered. For more information about reentering master keys in LPAR mode, see Appendix D, “PR/SM Considerations during Key Entry,” on page 403.

Note: If you used the Pass Phrase Initialization (PPINIT) utility to initialize your system, use the utility to reload the master keys. See "Steps for reinitializing a system" on page 99 for additional information.

When the cryptographic feature clears the master keys, reenter the same master keys by using these steps:

1. Check the status of the PKA callable services control if applicable. If it is enabled, use the Administrative Control Functions to disable it. See "Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls" on page 109 for details.
2. Retrieve the key parts, checksums, verification patterns, and hash patterns you used when you entered the master keys originally. These values should be stored in a secure place as specified in your enterprises security process.
3. Access the Master Key Entry panels and enter the master keys as described in "Steps for entering the first master key part" on page 117.
4. After you have entered the master keys, select option 2, KDS MANAGEMENT, from the primary menu. The Key Data Set Management panel appears. See Figure 29.

   To activate the master keys you just entered, you need to set them.
5. To set any master key, choose option 4 on the panel and press ENTER.



```
 CSFMKM10 ---------------- ICSF - Key Data Set Management  -------------
 OPTION ===>  4

 Enter the number of the desired option.

   1  CKDS MK MANAGEMENT - Perform Cryptographic Key Data Set (CKDS)
                           master key management functions
   2  PKDS MK MANAGEMENT-  Perform Public Key Data Set (PKDS)
                           master key management functions
   3  TKDS MK MANAGEMENT-  Perform PKCS #11 Token Data Set TKDS)
                           master key management functions
   4  SET MK            -  Set master keys

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


OPTION ===>
```

Figure 29. Selecting the Set Host Master Key Option on the Key Data Set Management panel

   When you select option 4, ICSF checks all coprocessors. If the value in any new master key register matches the active CKDS and PKDS, ICSF will transfer the master key from the new master key register to the current master key register. This process sets the master keys that matches the active CKDS and PKDS.

   When ICSF attempts to set the master keys, it displays a message on the top right of the Key Data Set Management panel. The message indicates either that the master key was successfully set or that an error prevented the completion of the set process.

   When you set the reentered master keys, the master keys that encipher the active CKDS and PKDS now exist. There is no need to refresh the CKDS or PKDS.
6. You can now change the master keys, if you choose to, for security reasons. Continue with "Changing the master keys" on page 140.

Initializing the key data sets at first-time startup

If running in a sysplex, see Chapter 10, "Running in a Sysplex Environment," on page 171.

A CKDS or PKDS is not required to use ICSF, but you can have a CKDS, a PKDS, or both. By defining and initializing a CKDS and PKDS, secure CCA symmetric and asymmetric key functions are available, and ICSF can be used to manage CCA key tokens that are stored in the CKDS and PKDS.

When you are using a CKDS, you must:
- Create a cryptographic key data set (CKDS).
- Enter a new DES-MK into all coprocessors (optional).
- Enter a new AES-MK into all coprocessors (optional).
- Initialize the CKDS.

When you are using a PKDS, you must:
- Create a public key data set (PKDS).
- Enter a new RSA-MK into all coprocessors (optional).
- Enter a new ECC-MK into all coprocessors (optional).
- Initialize the PKDS.

When you initialize the CKDS, ICSF creates a header record for the CKDS and sets any DES or AES master keys in the new master key registers. When you initialize the PKDS, ICSF creates a header record for the PKDS and sets any ECC or RSA master keys in the new master key registers.

Important: The master keys are loaded into the new master key register. The utilities that are described in this section require the master keys to be in the new master key register and not set. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. This option must not be used for a coprocessor or domain that is visible to ICSF. These utilities fail if the master key is not in the new master key register.

CKDS

A CKDS is not required in order to use ICSF. However, by defining and initializing a CKDS, secure CCA symmetric key functions are available, and ICSF can be used to manage CCA symmetric key tokens in the CKDS. When you initialize a CKDS, you can copy the disk copy of the CKDS to create other CKDSs for use on the system. You can also use a CKDS on another ICSF system if the system has the same master key value.

Note: Use of a CKDS on another system depends both upon where the CKDS was initialized and the cryptographic hardware type of the other system. At any time, you can read a different disk copy into storage. For information about how to read a disk copy into storage on a single system, see "Refreshing the key data sets" on page 134.

For a description of how to use the Master Key Entry panels to enter the master key, see "Steps for entering the first master key part" on page 117. For a description of how to use the TKE workstation to enter the master key, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Steps for initializing a CKDS

For information about initializing a CKDS in a sysplex environment, see Chapter 10, “Running in a Sysplex Environment,” on page 171.

There are three formats of the CKDS. You can use the following steps to initialize any format of CKDS.

- A fixed length record format supported by all releases of ICSF.
- A variable length record format supported by ICSF FMID HCR7780 and later releases.
- A common variable length record format supported by ICSF FMID HCR77A1 and later releases. This format is referred to as KDSR format.

To initialize the CKDS:

1. From the ICSF Primary Menu, select option 2, KDS MANAGEMENT.

```
HCR77C0 ------------- Integrated Cryptographic Service Facility ---------

OPTION ===>

Enter the number of the desired option.

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

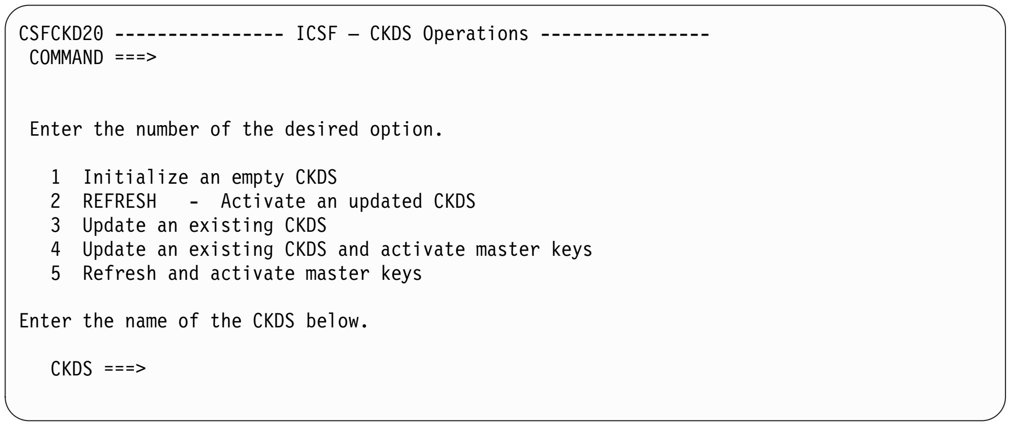
Figure 30. Selecting KDS Management on the ICSF primary menu panel

2. The Key Data Set Management panel appears. Select option 1, CKDS MK MANAGEMENT.

```
CSFMKM10 ---------------- ICSF – Key Data Set Management ----------------
OPTION ===>

Enter the number of the desired option.

  1  CKDS MK MANAGEMENT - Perform Cryptographic Key Data Set (CKDS)
                          functions including master key management
  2  PKDS MK MANAGEMENT - Perform Public Key Data Set (PKDS)
                          functions including master key management
  3  TKDS MK MANAGEMENT - Perform PKCS #11 Token Data Set (TKDS)
                          functions including master key management
  4  SET MK             - Set master key
```

Figure 31. Selecting CKDS MK MANAGEMENT on the Key Data Set Management panel

3. The CKDS Operations panel appears:



```
CSFCKD20 ---------------- ICSF – CKDS Operations ----------------
 COMMAND ===>


 Enter the number of the desired option.

   1  Initialize an empty CKDS
   2  REFRESH   -  Activate an updated CKDS
   3  Update an existing CKDS
   4  Update an existing CKDS and activate master keys
   5  Refresh and activate master keys

Enter the name of the CKDS below.

   CKDS ===>
```

Figure 32. The CKDS Operations panel

In the CKDS field, enter the name of the empty VSAM data set that was created to use as the disk copy of the CKDS.

- The name you enter can be the same name that is specified in the CKDSN keyword option in the installation options data set. You can also initialize a data set that might serve as a backup. For information about creating a CKDS and specifying the CKDS name in the installation options data set, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

4. Select option 1, Initialize an empty CKDS. This will activate the master keys.

   ICSF creates the header record in the disk copy of the CKDS and refreshes the CKDS. Next, ICSF sets the DES or AES master key, if any. ICSF then adds the required system key to the CKDS and refreshes the CKDS. When ICSF completes all these steps, the message INITIALIZATION COMPLETE appears. If you did not enter a master key into the new master key register previously, the message NMK REGISTER NOT FULL appears and the initialization process ends. You must enter a master key into the new master key register to initialize the CKDS.

   Note: If any part of the option 1 fails, you must delete the CKDS and start over. If the failure occurs when one of the master keys has been set and prior to the system key being created, you will need to reset the master key.

   When you complete the entire process, a CKDS and zero or more master keys exist on your system. You can now generate keys using functions like the key generate callable service and the key generator utility program (KGUP) or convert PCF keys to ICSF keys using the PCF CKDS conversion program. ICSF services use the keys to perform the cryptographic functions you request.

PKDS

A PKDS is not required in order to use ICSF. However, by defining and initializing a PKDS, secure CCA asymmetric key functions are available, and ICSF can be used to manage CCA asymmetric key tokens in the PKDS. When you initialize a PKDS, you can copy the disk copy of the PKDS to create other PKDSs for use on the system. You can also use a PKDS on another ICSF system if the system has the same master key value.

Note: Use of a PKDS on another system depends both upon where the PKDS was initialized and the cryptographic hardware type of the other system. At any time, you can read a different disk copy into storage. For information about how to read a disk copy into storage on a single system, see "Refreshing the key data sets" on page 134 for additional information.

For a description of how to use the Master Key Entry panels to enter the master key, see "Steps for entering the first master key part" on page 117. For a description of how to use the TKE workstation to enter the master key, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Steps for initializing the PKDS

For information about initializing a PKDS in a sysplex environment, see Chapter 10, "Running in a Sysplex Environment," on page 171.

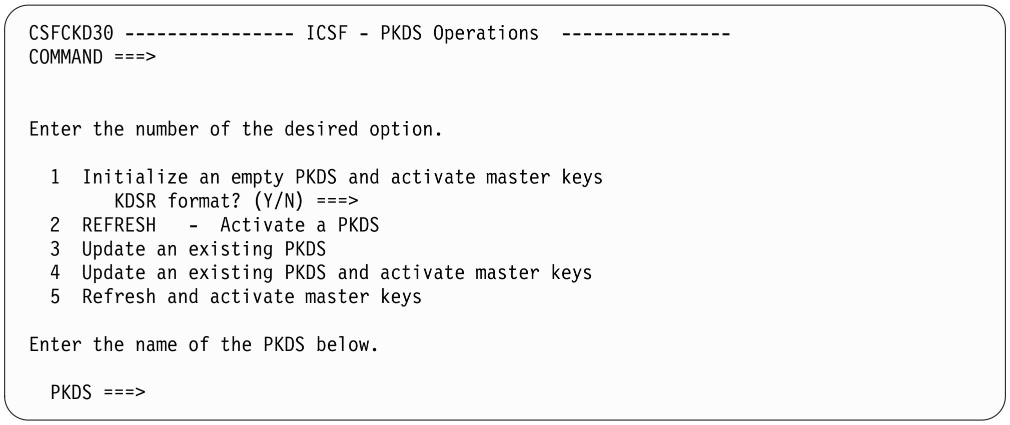
There are two formats of the PKDS. You can use the following steps to initialize any format of PKDS.

- PKDS record format supported by all releases of ICSF.
- Common record format supported by ICSF FMID HCR77A1 and later releases. This format is referred to as KDSR format.

To initialize the PKDS:

1. Select option 2, KDS MANAGEMENT, on the "ICSF Primary Menu panel" on page 387.

   The "CSFMKM10 — Key Data Set Management panel" on page 390 appears.
2. Select option 2, for PKDS MK MANAGEMENT. The "CSFMKM30 — PKDS Management panel" on page 391 appears.
3. Select option 1, INIT/REFRESH/UPDATE PKDS and the Initialize a PKDS panel appears. See Figure 33.



```
CSFCKD30 ---------------- ICSF - PKDS Operations  ----------------
COMMAND ===>


Enter the number of the desired option.

  1  Initialize an empty PKDS and activate master keys
        KDSR format? (Y/N) ===>
  2  REFRESH   -  Activate a PKDS
  3  Update an existing PKDS
  4  Update an existing PKDS and activate master keys
  5  Refresh and activate master keys

Enter the name of the PKDS below.

  PKDS ===>
```

Figure 33. ICSF Initialize/Refresh a PKDS Panel

4. In the PKDS field, enter the name of the empty VSAM data set that was created to use as the disk copy of the PKDS.
   - The name you enter can be the same name that is specified in the PKDSN keyword option in the installation options data set. You can also initialize a data set that might serve as a backup. For information about creating a PKDS and specifying the PKDS name in the installation options data set, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.
   - In the KDSR format? field, enter Y if you want the PKDS to use the common record format. Enter N if you want to use the old format.
5. Select option 1, Initialize an empty PKDS and activate master keys and then press ENTER.

ICSF creates the header record in the disk copy of the PKDS. Next, ICSF sets the ECC or RSA master key, if any. ICSF then adds the required system key to the PKDS and refreshes the PKDS. When ICSF completes all these steps, the message INITIALIZATION COMPLETE appears. If you did not enter a master key into the new master key register previously, the message NMK REGISTER NOT FULL appears and the initialization process ends. You must enter a master key into the new master key register to initialize the PKDS.

Note: If any part of the option 1 fails, you must delete the PKDS and start over. If the failure occurs when one of the master keys has been set and prior to the system key being created, you will need to reset the master key.

When you complete the entire process, a PKDS and zero or more master keys exist on your system. You can now generate keys using functions like the PKA key generate callable service. ICSF services use the keys to perform the cryptographic functions you request.

Updating the key data sets with additional master keys

If running in a sysplex, see Chapter 10, “Running in a Sysplex Environment,” on page 171.

When you add a new master key to your system for your applications, you need to update the key date set so keys can be added to the key data set and ICSF knows the new master key is required for coprocessor activation.

Important: The master keys are loaded into the new master key register. The utilities described in this section require the master keys to be in the new master key register and not set. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. This option must not be used for a coprocessor or domain that is visible to ICSF. These utilities will fail if the master key is not in the new master key register.

CKDS

You can add the AES master key to a CKDS that was initialized with only the DES master key. It is also possible to add the DES master key to a CKDS that was initialized with only the AES master key.

There are three options for updating the CKDS on the CKDS Operations panel.

- Option 3, 'Update an existing CKDS', adds the missing master key verification pattern to the CKDS header record. The CKDS will not become the active CKDS and the master key will not be set. Use this option if you have more than one CKDS to update as a CKDS cannot be updated if the new master key register is empty. The last CKDS should be processed using option 4.
- Option 4, 'Update an existing CKDS and activate master keys', adds the missing master key verification pattern, makes the specified CKDS the active CKDS, and sets the master keys. This option should be used if you have only one CKDS to update or to update the last of your CKDSs after all other CKDSs have been updated using option 3, 'Update an existing CKDS'.
- Option 5, 'Refresh and activate master keys', makes the specified CKDS the active CKDS and sets the master keys. This option is used on other LPARs after the CKDS has been updated with the missing master key to activate the CKDS and to set the master keys.

These are the steps to update the CKDS:

1. Load the new master key register for the missing master key by using the master key entry panels or by using the TKE workstation. The missing master key must be loaded on all active coprocessors.
2. From the “ICSF Primary Menu panel” on page 387, select option 2, KDS MANAGEMENT.
3. From the “CSFMKM10 — Key Data Set Management panel” on page 390, select option 1 for CKDS MK MANAGEMENT.
4. The “CSFMKM20 — CKDS Management panel” on page 390 appears, select option 1, CKDS OPERATIONS.
5. The “CSFCKD20 — CKDS Operations panel” on page 388 appears. If updating multiple CKDS, for all but the last CKDS, choose option 3, Update an existing CKDS and press ENTER. ICSF will check the status of the new master key registers and that the master key verification pattern of the master key is written to the CKDS header record.

   Note: All the CKDSs that you wish to update should be processed prior to going to the next step.
6. In the CKDS field, enter the name of an existing, initialized CKDS.
7. If updating the last or only CKDS, choose option 4, 'Update an existing CKDS and activate master keys' and press ENTER. ICSF will check the status of the new master key registers and that the master key verification pattern of the master key is written to the CKDS header record. The CKDS will become the active CKDS and sets the master key.

PKDS

On systems that support the ECC master key, you can add the ECC master key to any existing PKDS. It is also possible to add the RSA master key to a PKDS that was initialized with only the ECC master key.

There are three options for updating the PKDS on the PKDS Operations panel.

- Option 3, 'Update an existing PKDS', adds the missing master key verification pattern to the PKDS header record. The PKDS will not become the active PKDS and the master key will not be set. Use this option if you have more than one PKDS to update as a PKDS cannot be updated if the new master key register is empty. The last PKDS should be processed using option 4, 'Update an existing PKDS and activate master keys'.
- Option 4, 'Update an existing PKDS and activate master keys', adds the missing master key verification pattern, makes the specified PKDS the active PKDS, and sets the master keys. This option should be used if you have only one PKDS to update or to update the last of your PKDSs after all other PKDSs have been updated using option 3.
- Option 5, 'Refresh and activate master keys', makes the specified PKDS the active PKDS and sets the master keys. This option is used on other LPARs after the PKDS has been updated with the missing master key to activate the PKDS and to set the master keys.

These are the steps to update the PKDS:

1. Load the new master key register for the missing master key by using the master key entry panels or by using the TKE workstation. The missing master key must be loaded on all active coprocessors.
2. From the “ICSF Primary Menu panel” on page 387, select option 2, KDS MANAGEMENT.

3. From the “CSFMKM10 — Key Data Set Management panel” on page 390, select option 2 for PKDS MK MANAGEMENT.
4. The “CSFMKM30 — PKDS Management panel” on page 391 appears, select option 1, PKDS OPERATIONS.
5. The “CSFCKD30 — PKDS Operations panel” on page 388 appears. In the PKDS field, enter the name of an existing, initialized PKDS.
6. If updating multiple PKDS, for all but the last PKDS, choose option 3, Update an existing PKDS, and press ENTER. ICSF will check the status of the new master key registers and the master key verification pattern of the master key is written to the PKDS header record.

   Note: All the PKDSs that you wish to update should be processed prior to going to the next step.
7. If updating the last or only PKDS, choose option 4, Update an existing PKDS and activate master keys, and press ENTER. ICSF will check the status of the new master key registers and that the master key verification pattern of the master key is written to the PKDS header record. The PKDS will become the active PKDS and sets the master key.

Refreshing the key data sets

You can refresh the in-storage key data set with an updated or different disk copy of the key data set. You can refresh the key data sets at any time without disrupting cryptographic functions.

When you initialize a CKDS for the first time, you can copy the disk copy of the CKDS to create other CKDSs for the system. You can use KGUP to add and update any of the disk copies on your system. You can use the dynamic CKDS update callable services to add or update the disk copy of the current in-storage CKDS. For information about using KGUP, see Chapter 11, “Managing Cryptographic Keys Using the Key Generator Utility Program,” on page 185. For information on using the dynamic CKDS callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*z/OS Cryptographic Services ICSF Application Programmer's Guide.

When you initialize a PKDS for the first time, you can make disk copies to create other PKDSs for the system. You can use the dynamic PKDS update callable services to add or update the disk copy of the current in-storage PKDS. For information on using the dynamic PKDS callable services, refer to the *z/OS Cryptographic Services ICSF Application Programmer's Guide*z/OS Cryptographic Services ICSF Application Programmer's Guide. You can refresh the in-storage PKDS with an updated or different disk copy of the PKDS by using these steps. You can refresh the PKDS at any time without disrupting cryptographic functions.

The coordinated refresh utilities will refresh the key data set for all members in the sysplex sharing that key data set. The local refresh utilities will only refresh the key data set on the system where the utility is invoked.

If you are running either a stand-alone system or a sysplex environment where all members of the systems are using ICSF FMID HCR7790 or later, you may be able to perform a coordinated CKDS refresh. The coordinated CKDS refresh operation simplifies CKDS administration by automating steps from the local CKDS refresh procedure and allowing the refresh to be initiated from one member. Coordinated CKDS refresh is carried out for all members in the sysplex sharing the same active

CKDS. If you are in a single system environment, coordinated CKDS refresh can still be used to automate the manual steps of a local CKDS refresh.

If you are running either a stand-alone system or a sysplex environment where all members of the sysplex are using ICSF FMID HCR77A0 or later, you may be able to perform a coordinated PKDS refresh. The coordinated PKDS refresh operation simplifies PKDS administration by automating steps from the local PKDS refresh procedure and allowing the refresh to be initiated from a one member of the sysplex. Coordinated PKDS refresh is carried out for all members in the sysplex sharing the same active PKDS. If you are in a single system environment, coordinated PKDS refresh can still be used to automate the manual steps of a local PKDS refresh.

On systems where there are active coprocessors (the MKVP in the key data set matches the current master key register), the MKVPs in the header of the new key data set must be the same as the active key data set. ICSF will not allow a system to refresh to a key data set that will cause any active coprocessors to become inactive.

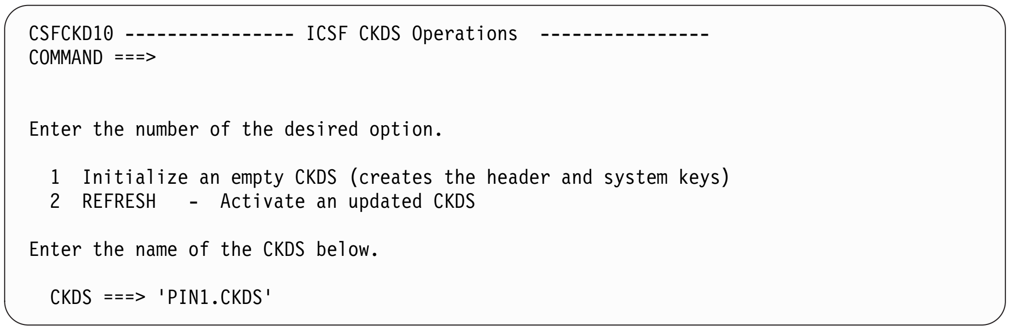
Performing a local CKDS refresh

You use the CKDS utility program, CSFEUTIL, to invoke a local refresh of the CKDS from a batch job. See Chapter 17, “Using the ICSF Utility Program CSFEUTIL,” on page 343.

You can refresh the in-storage CKDS with an updated or different disk copy of the CKDS by using these steps. You can refresh the CKDS at any time without disrupting cryptographic functions.

Note: Prior to refreshing a CKDS, consider temporarily disallowing dynamic CKDS update services. For more information, see “Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls” on page 109.

1. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel: “ICSF Primary Menu panel” on page 387
2. Enter option 1, CKDS MK MANAGEMENT and the CKDS Management panel appears: “CSFMKM10 — Key Data Set Management panel” on page 390
3. Enter option 1, CKDS Operations to access the CKDS Operations panel.



```
CSFCKD10 ---------------- ICSF CKDS Operations  ----------------
COMMAND ===>


Enter the number of the desired option.

  1  Initialize an empty CKDS (creates the header and system keys)
  2  REFRESH   -  Activate an updated CKDS

Enter the name of the CKDS below.

  CKDS ===> 'PIN1.CKDS'
```

Figure 34. Selecting the Refresh Option on the ICSF Initialize a CKDS Panel

4. In the CKDS field, specify the name of the disk copy of the CKDS that you want ICSF to read into storage.
5. Choose option 2, REFRESH, and press ENTER.

ICSF places the disk copy of the specified CKDS into storage. A REFRESH does not disrupt any applications that are running on ICSF. A message that states that the CKDS was refreshed appears on the right of the top line on the panel.

If you have CKDS record authentication enabled, ICSF performs a MAC verification on each record in the CKDS. When ICSF reads the CKDS into storage, it performs a MAC verification on each record in the CKDS. If a record fails the MAC verification, ICSF sends a message that gives the key label and type to the z/OS system security console. You can then use either KGUP or the dynamic CKDS update services to delete the record from the CKDS. Any other attempts to access a record that has failed MAC verification results in a return code and reason code that indicate that the MAC is not valid.

6. Press END to return to the Primary Menu panel.

Note: You can use either a KGUP panel or a utility program, instead of the CKDS panel, to perform a local CKDS refresh. For information about these other methods, see "Performing a local CKDS refresh" on page 135 with KGUP.

Performing a coordinated CKDS refresh

A coordinated CKDS refresh may be performed on a single instance of ICSF, on a single-system sysplex, or on a multi-system sysplex. The coordinated refresh operation is initiated from a single ICSF instance and then carried out across all other sysplex members sharing the same active CKDS. This results in the in-storage copy of the CKDS being updated for all ICSF instances in the sysplex that share the same active CKDS as the initiator.

To perform a coordinated CKDS refresh, all members of the sysplex (including sysplex members that are not configured with the same active CKDS) must be at the ICSF FMID HCR7790 release or later.

A coordinated CKDS refresh can be done by writing an application to invoke the Coordinated KDS Administration (CSFCRC) callable service. For information on this callable service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

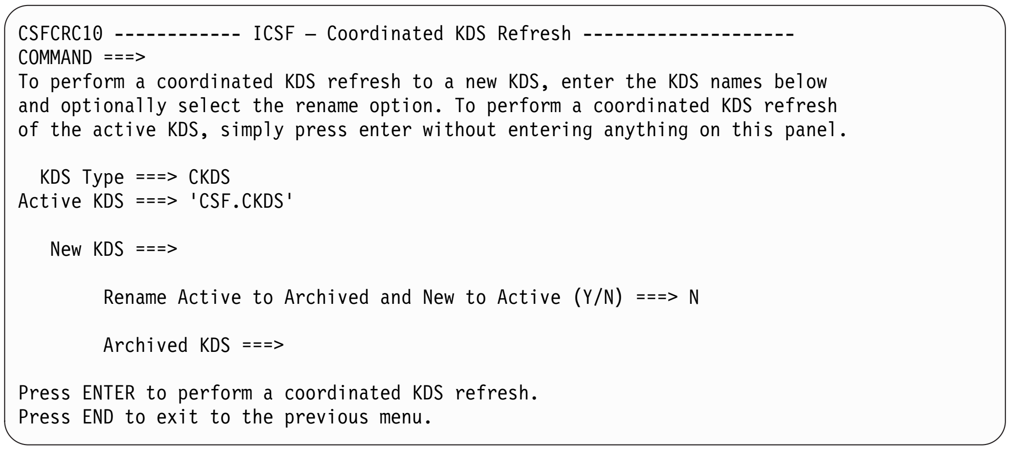
Before performing a refresh, you should consider temporarily disallowing dynamic CKDS updates on all sysplex members for the CKDS you are processing. For information on disabling dynamic CKDS updates, See "Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls" on page 109.

If you are performing a refresh to a new CKDS, the new CKDS must be allocated, must not be empty, and must be enciphered with the current master key or keys. You will optionally be able to use the archive option for renaming the current CKDS to an archive name and the new CKDS to the active CKDS name. The archive data set name must not be allocated or exist on the system prior to performing the refresh.

To perform a coordinated CKDS refresh:

1. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel.
2. Enter option 1, CKDS MK MANAGEMENT, and the CKDS Management panel appears: "CSFMKM20 — CKDS Management panel" on page 390

3. Enter option 4, COORDINATED CKDS REFRESH, and the Coordinated KDS Refresh panel appears:



```
CSFCRC10 ------------ ICSF – Coordinated KDS Refresh --------------------
COMMAND ===>
To perform a coordinated KDS refresh to a new KDS, enter the KDS names below
and optionally select the rename option. To perform a coordinated KDS refresh
of the active KDS, simply press enter without entering anything on this panel.

  KDS Type ===> CKDS
Active KDS ===> 'CSF.CKDS'

   New KDS ===>

        Rename Active to Archived and New to Active (Y/N) ===> N

        Archived KDS ===>

Press ENTER to perform a coordinated KDS refresh.
Press END to exit to the previous menu.
```

Figure 35. The Coordinated KDS Refresh panel

The active CKDS name is displayed in the **Active KDS** field. You can use this panel to refresh to a new CKDS or to refresh the active CKDS.

To refresh to a new CKDS:

a. Enter the name of the new CKDS in the **New KDS** field. This data set must be allocated, not empty, and enciphered under the current master key or keys.

b. Optionally, the rename option may be used to have the current CKDS renamed to an archive name and the new CKDS renamed to the active CKDS name. The rename option simplifies CKDS administration by removing the need to update the ICSF Installation Options Data Set with the name of the new data set after the coordinated CKDS refresh to a new CKDS completes.

   If you would like to have the new CKDS renamed to match the name of the current active CKDS:

   1) Type Y in the **Rename Active to Archived and the New to Active (Y / N)** field.

   2) Enter the name under which the currently active CKDS will be archived in the **Archived KDS**field. The archive CKDS name must not be allocated and must not exist on the system prior to performing the coordinated refresh to a new data set.

   If you do not want to have the new CKDS renamed to match the name of the current active CKDS, type N in the **Rename Active to Archived and the New to Active (Y / N)** field. Remember to change the name of the KDS in the Installation Options Data Set as described in *z/OS Cryptographic Services ICSF System Programmer's Guide*. The CKDS name must be changed in each cluster member's Installation Options Data Set after the coordinated CKDS refresh function completes successfully. If the Installation Options Data Set is updated with a new CKDS name and the coordinated CKDS refresh function fails, ICSF might be configured with an invalid CKDS the next time it is restarted.

c. Press ENTER to begin the coordinated refresh.

   To refresh the active CKDS, no input is required on the panel and will be ignored if entered.

1) Verify that the **Active KDS** field shows the name of the active CKDS. ICSF should have filled in this field automatically.
2) Press ENTER to begin the coordinated refresh.

4. A confirmation panel is displayed, prompting you to verify that you want to continue with the coordinated refresh. Verify that the information on this confirmation panel is correct. If it is, type Y in the confirmation field provided and press ENTER. The Coordinated KDS Refresh will then start processing.
5. Verify the dialog results and correct any indicated failures or unexpected results. See "Recovering from a coordinated administration failure" on page 150 for additional information.

Performing a local PKDS refresh

You use the PKDS utility program, CSFPUTIL, to invoke a local refresh of the PKDS from a batch job. See Chapter 18, "Using the ICSF Utility Program CSFPUTIL," on page 353.

Note: Prior to refreshing a PKDS, consider temporarily disallowing dynamic PKDS update services. For more information, see "Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls" on page 109.

1. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel: "ICSF Primary Menu panel" on page 387

   The "CSFMKM10 — Key Data Set Management panel" on page 390 appears.
2. Enter option 2, PKDS MK MANAGEMENT and the PKDS Management panel appears: "CSFMKM30 — PKDS Management panel" on page 391
3. Enter option 1, PKDS Operations to access the PKDS Operations panel appears: "CSFCKD30 — PKDS Operations panel" on page 388
4. In the PKDS field, specify the name of the disk copy of the PKDS that you want ICSF to read into storage. ICSF places the disk copy of the specified PKDS into storage. A message that states that the PKDS was refreshed appears on the right of the top line on the panel.
5. Choose option 2, REFRESH, and press ENTER.
6. Press END to return to the Primary Menu panel.

ICSF places the disk copy of the specified PKDS into storage. A REFRESH does not disrupt any applications that are running on ICSF. A message that states that the PKDS was refreshed appears on the right of the top line on the panel.

Performing a coordinated PKDS refresh

A coordinated PKDS refresh may be performed on a single instance of ICSF, on a single-system sysplex, or on a multi-system sysplex. The coordinated refresh operation is initiated from a single ICSF instance and then carried out across all other sysplex members sharing the same active PKDS. This results in the in-storage copy of the PKDS being updated for all ICSF instances in the sysplex that share the same active PKDS as the initiator.

To perform a coordinated PKDS refresh, all members of the sysplex (including sysplex members that are not configured with the same active PKDS) must be at the ICSF FMID HCR77A0 release or later.

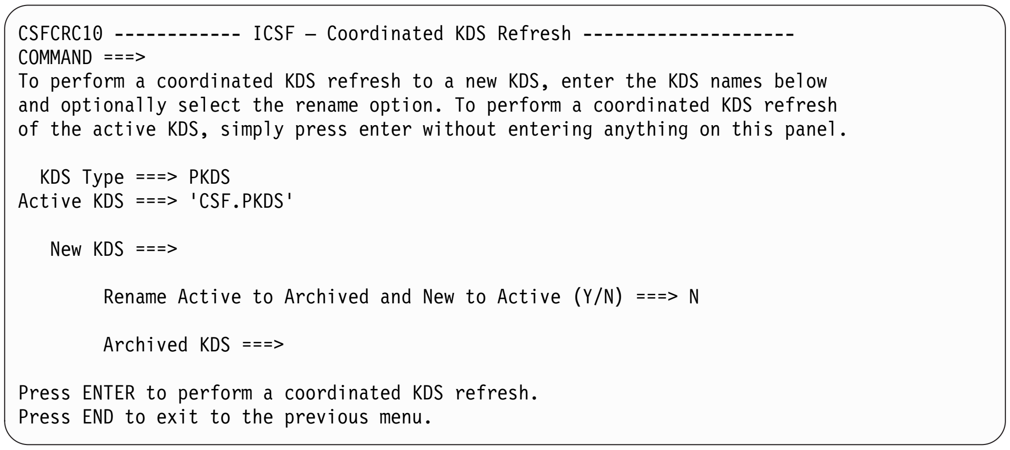
A coordinated PKDS refresh can be done by writing an application to invoke the Coordinated KDS Administration (CSFCRC) callable service. For information on this callable service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Before performing a refresh, you should consider temporarily disallowing dynamic PKDS updates on all sysplex members for the PKDS you are processing. For information on disabling dynamic PKDS updates, See "Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls" on page 109.

If you are performing a refresh to a new PKDS, the new PKDS must be allocated, must not be empty, and must be enciphered with the current master key or keys. You will optionally be able to use the archive option for renaming the current PKDS to an archive name and the new PKDS to the active PKDS name. The archive data set name must not be allocated or exist on the system prior to performing the refresh.

To perform a coordinated PKDS refresh:

1. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel.
2. Enter option 2, PKDS MK MANAGEMENT, and the PKDS Management panel appears: "CSFMKM30 — PKDS Management panel" on page 391
3. Enter option 4, COORDINATED PKDS REFRESH, and the Coordinated KDS Refresh panel appears:



```
CSFCRC10 ------------ ICSF - Coordinated KDS Refresh --------------------
COMMAND ===>
To perform a coordinated KDS refresh to a new KDS, enter the KDS names below
and optionally select the rename option. To perform a coordinated KDS refresh
of the active KDS, simply press enter without entering anything on this panel.

  KDS Type ===> PKDS
Active KDS ===> 'CSF.PKDS'

   New KDS ===>

        Rename Active to Archived and New to Active (Y/N) ===> N

        Archived KDS ===>

Press ENTER to perform a coordinated KDS refresh.
Press END to exit to the previous menu.
```

Figure 36. The Coordinated KDS Refresh panel

The active PKDS name is displayed in the **Active KDS** field. You can use this panel to refresh to a new PKDS or to refresh the active PKDS.

To refresh to a new PKDS:

a. Enter the name of the new PKDS in the **New KDS** field. This data set must be allocated, not empty, and enciphered under the current master key or keys.
b. Optionally, the rename option may be used to have the current PKDS renamed to an archive name and the new PKDS renamed to the active PKDS name. The rename option simplifies PKDS administration by removing the need to update the ICSF Installation Options Data Set with the name of the new data set after the coordinated PKDS refresh to a new PKDS completes.

If you would like to have the new PKDS renamed to match the name of the current active PKDS:

1) Type Y in the **Rename Active to Archived and the New to Active (Y / N)** field.

2) Enter the name under which the currently active PKDS will be archived in the **Archived KDS**field. The archive PKDS name must not be allocated and must not exist on the system prior to performing the coordinated refresh to a new data set.

If you do not want to have the new PKDS renamed to match the name of the current active PKDS, type N in the **Rename Active to Archived and the New to Active (Y / N)** field. Remember to change the name of the KDS in the Installation Options Data Set as described in *z/OS Cryptographic Services ICSF System Programmer's Guide*. The PKDS name must be changed in each cluster member's Installation Options Data Set after the coordinated PKDS refresh function completes successfully. If the Installation Options Data Set is updated with a new PKDS name and the coordinated PKDS refresh function fails, ICSF might be configured with an invalid PKDS the next time it is restarted.

c. Press ENTER to begin the coordinated refresh.

To refresh the active PKDS, no input is required on the panel and will be ignored if entered.

1) Verify that the **Active KDS** field shows the name of the active PKDS. ICSF should have filled in this field automatically.

2) Press ENTER to begin the coordinated refresh.

4. A confirmation panel is displayed, prompting you to verify that you want to continue with the coordinated refresh. Verify that the information on this confirmation panel is correct. If it is, type Y in the confirmation field provided and press ENTER. The Coordinated KDS Refresh will then start processing.
5. Verify the dialog results and correct any indicated failures or unexpected results. See "Recovering from a coordinated administration failure" on page 150 for additional information.

Changing the master keys

For security reasons, your installation should change the master keys periodically. In addition, if the master keys have been cleared, you might also want to change the master keys when you reenter the cleared master keys.

There are three main steps that are involved in performing a master key change:

1. Enter the master key parts by using the ICSF Master Key Entry or the TKE workstation.
2. Reencipher the key data sets under the new master keys. This fills an empty VSAM data set with the reenciphered keys and makes the data set the new key data set. This new reenciphered key data set is a disk copy.
3. Change the new master keys and activate the reenciphered key data sets.

Note:

- DES and AES master keys can be changed separately or together.
- RSA and ECC master keys can be changed separately or together.

If your system is using multiple coprocessors, they must have the same master key or keys. When you load a new master key or keys in one coprocessor, you should load the same new master key or keys in the other coprocessors. Therefore, to reencipher a key data set under a new master key, the new master key registers in all coprocessors must contain the same value.

Important: The master keys are loaded into the new master key register. The utilities that are described in this section require the master keys to be in the new master key register and not set. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. This option must not be used for a coprocessor or domain that is visible to ICSF. Setting the master keys should be done by using ICSF utilities so ICSF knows that the master keys have changed. Changing the master keys without using ICSF may cause an outage.

When using the local reencipher KDS and change master key utilities, each step is performed separately and only on the system where the utility is invoked.

When using the coordinated change master key utility after the new master key is loaded, the remaining steps will be done by the utility across all members of the sysplex that share the same key data set. Additional function for renaming and archiving the key data set is available.

The coordinated change master key utility combines the CKDS or PKDS reencipher and set master key steps for both single system environments and sysplex environments. Additionally, when in a sysplex environment, the coordinated change master key utility coordinates across all sysplex members that share the same active CKDS or PKDS. This removes the need to perform manual steps on each system that shares the same CKDS or PKDS, including bringing the disk copy of the reenciphered CKDS or PKDS into storage. Be aware that this utility changes the master keys for all systems in the sysplex that share the same active CKDS or PKDS as the member who initiates the procedure.

If you are running either a stand-alone system or a sysplex environment where all members of the systems are using ICSF FMID HCR7790 or later, you may be able to perform a coordinated CKDS change master key. All members of the sysplex (including any sysplex members that are not using the same active CKDS) must be at ICSF FMID HCR7790 or later. ICSF on all systems in the sysplex must be running in noncompatibility mode.

If you are running either a stand-alone system or a sysplex environment where all members of the systems are using ICSF FMID HCR77A0 or later, you may be able to perform a coordinated PKDS change master key. All members of the sysplex (including any sysplex members that are not using the same active PKDS) must be at ICSF FMID HCR77A0 or later.

Note: Do not use the coordinated KDS change master key procedure to reencipher archived or backup copies of the CKDS or PKDS that are not currently active. Use it to only reencipher the active CKDS or PKDS.

If an error reenciphering a key in a key data set that causes the utility to fail, a CSFC0316 message is generated specifying the label for the problem record. You can use the Key Check utility to check the keys in a key data set before reenciphering the key data set.

Key check utility

Note: The key check utility is available on IBM z13 and later servers.

The key check utility validates each key in the key data set in the same manner as the KDS reencipher utility without reenciphering the key. The utility can be run prior to reenciphering on a key data set to ensure the KDS reencipher and change master key utilities will complete successfully. The utility checks the active key data sets. The results of the check are written to the ICSF joblog.

The key check utility can be invoked from the CKDS Management and PKDS Management panels or by writing an application to invoke the ICSF Multi-Purpose Service (CSFMPS) callable service. For more information, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Steps for checking the CKDS

1. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel: "ICSF Primary Menu panel" on page 387 to access the Key Data Set Management panel: "CSFMKM10 — Key Data Set Management panel" on page 390
2. Enter option 1, CKDS MK MANAGEMENT and the CKDS Management panel appears: "CSFMKM20 — CKDS Management panel" on page 390
3. Enter option 7 for CKDS KEY CHECK to run the utility on the active CKDS.

   The panel message will indicate if the check was successful or if there are errors. CSFM661I messages are written to the ICSF joblog with warnings and errors.

Steps for checking the PKDS

1. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel: "ICSF Primary Menu panel" on page 387 to access the Key Data Set Management panel: "CSFMKM10 — Key Data Set Management panel" on page 390
2. Enter option 2, PKDS MK MANAGEMENT and the PKDS Management panel appears: "CSFMKM30 — PKDS Management panel" on page 391
3. Enter option 7 for PKDS KEY CHECK to run the utility on the active PKDS.

   The panel message will indicate if the check was successful or if there are errors. CSFM661I messages are written to the ICSF joblog with warnings and errors.

Symmetric master keys and the CKDS

The procedure you need to follow for changing the symmetric master keys will differ depending on factors such as your system's compatibility mode. Although the details of the various procedures differ, they guide you through performing the same significant actions. To change the symmetric keys, you need to:

1. Enter the master key parts into the new master key registers.
2. Reencipher the CKDS under the new master key.
3. Change the symmetric master keys and make the reenciphered CKDS the active CKDS.

Because this procedure branches into different instructions based on whether ICSF is running in noncompatibility, compatibility, or co-existence mode, you should first understand the following background information on these modes before referring to and performing the procedure.

ICSF runs in noncompatibility, compatibility, or co-existence mode with the IBM cryptographic products, and Programmed Cryptographic Facility (PCF). You specify which mode ICSF runs in by using an installation option. For a description of the modes and how to specify an installation option, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

In noncompatibility mode, ICSF allows you to change the master key with continuous operations. Therefore, applications can continue to run without disruption. However, when ICSF is in compatibility mode or co-existence mode, you should use a different procedure to activate the changed master key. This is to ensure that no application is holding an internal token with the wrong master key.

In all three modes, you enter the new master key and reencipher the disk copy of the CKDS under the new master key using the master key panels. In noncompatibility mode, you then activate the new master key and refresh the in-storage copy of the CKDS with the disk copy using the master key panels or a utility program.

In compatibility mode and coexistence mode however, activating the new master key and refreshing the in-storage copy of the CKDS does not reencipher internal key tokens under the new master key. ICSF applications that are holding internal key tokens which have been enciphered under the wrong master key will fail with a warning message. Applications that use the PCF macros, run with no warning message and produce erroneous results.

If you have a cryptographic feature installed, when you start ICSF, you must go to the Key Data Set Management panel and do a set master key (option 4, SET MK). This will change the master keys of all the cryptographic features.

A re-IPL ensures that a program does not access a cryptographic service that uses a key that is encrypted under a different master key. If a program is using an operational key, the program should either re-create or reimport the key, or generate a new key.

If a re-IPL is not practical in your installation, you can use this alternative method. Stop all cryptographic applications, especially those using PCF macros, when activating the new master key and refreshing the in-storage copy of the CKDS. This eliminates all operational keys that are encrypted under the current master key. When you start ICSF again, applications using an operational key can either re-create or reimport the key.

Steps for reenciphering the CKDS and performing a local symmetric master key change

If running in a sysplex, see Chapter 10, “Running in a Sysplex Environment,” on page 171.

Note: Prior to reenciphering a CKDS, consider temporarily disallowing dynamic CKDS update services. For more information, see “Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls” on page 109.

Before beginning this procedure, you must:

- Enter the key parts of the new master key that you want to replace the current master key into all coprocessors on your system. For information about how to do this procedure, see “Entering master key parts” on page 111. The new master key register must be full when you change the master key.
- Create a new VSAM data set in which the reenciphered keys will be placed to create the new reenciphered CKDS. This data set must be allocated and empty and must contain the same data set attributes as the active CKDS. For more information about defining a CKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Note: To reencipher the CKDS, your TSO region size must be large enough to load the CKDS into memory. The CKDS load operation alone requires #records * Max LRECL bytes. Your TSO region size must be large enough to accommodate this in addition to any other memory required by your TSO user.

You can use the ICSF utility panels or a utility program to reencipher the CKDSs and change the master key. See Chapter 17, "Using the ICSF Utility Program CSFEUTIL," on page 343 for instructions on how to use the utility program to reencipher a disk copy of a CKDS and to change a master key.

To reencipher the CKDS and change the master key using the ICSF panels:

1. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel: "ICSF Primary Menu panel" on page 387

   The "CSFMKM10 — Key Data Set Management panel" on page 390 appears.
2. Enter option 1, CKDS MK MANAGEMENT and the CKDS Management panel appears: "CSFMKM20 — CKDS Management panel" on page 390
3. Select option 2, REENCIPHER CKDS, on the "CSFMKM20 — CKDS Management panel" on page 390, and press ENTER.

   When you change the master key, you must first reencipher the disk copy of the CKDS under the new master key.

   Note: If your system is using multiple coprocessors, they must have the same master key. When you change the master key in one coprocessor, you should change the master key in the other coprocessors. Therefore, to reencipher a CKDS under a new master key, the new master key registers in all coprocessors must contain the same value.
4. The "CSFCMK10 — Reencipher CKDS panel" on page 388 appears. In the Input CKDS field, enter the name of the CKDS that you want to reencipher. In the Output CKDS field, enter the name of the data set in which you want to place the reenciphered keys.

   Reenciphering the disk copy of the CKDS does not affect the in-storage copy of the CKDS. On this panel, you are working with only a disk copy of the CKDS.
5. Press ENTER to reencipher the input CKDS entries and place them into the output CKDS.

   The message REENCIPHER SUCCESSFUL appears on the top right of the panel if the reencipher succeeds.

   Note: If the operation fails with a return and reason code, see section "Return and reason codes for the CSFEUTIL program" on page 345 or the *z/OS Cryptographic Services ICSF Application Programmer's Guide* Appendix A for a description of the failing return and reason codes.
6. If you have more than one CKDS on a disk, specify the information and press ENTER as many times as you need to reencipher all of them. Reencipher all your disk copies at this time. When you have reenciphered all the disk copies of the CKDS, you are ready to change the master key.
7. Press END to return to the "CSFMKM20 — CKDS Management panel" on page 390.

   Performing a local symmetric master key change involves refreshing the in-storage copy of the CKDS with a disk copy and activating the new master key.

8. If you are running in PCF compatibility or co-existence mode, do not select option 3, the Change option. To activate the changed master key when running in compatibility or co-existence mode, you need to re-IPL z/OS and start ICSF. When you re-IPL z/OS and start ICSF, you activate the changed master key and refresh the in-storage CKDS.
9. If you are running in noncompatibility mode, to change the master key, select option 3, CHANGE SYM MK, on the CKDS Management panel: "CSFMKM20 — CKDS Management panel" on page 390 The Change Master Key panel appears: "CSFCMK20 — Change Master Key panel" on page 389
10. In the New CKDS field, enter the name of the disk copy of the CKDS that you want ICSF to place in storage.

    You should have already reenciphered the disk copy of the CKDS under the new master key. The last CKDS name that you specified in the Output CKDS field on the "CSFCMK10 — Reencipher CKDS panel" on page 388 automatically appears in this field.
11. Press ENTER.

    ICSF loads the data set into storage where it becomes operational on the system. ICSF also places the new master key into the master key register so it becomes active.

    When you press ENTER, ICSF attempts to change the master key. It displays a message on the top right of the panel. The message indicates either that the master key was changed successfully or that an error occurred that prevented the successful completion of the change process. For example, if you indicate a data set that is not reenciphered under the new master key, an error message displays and the master key is not changed.
12. When changing the master key, remember to change the name of the CKDS in the Installation Options Data Set.

Asymmetric master keys and the PKDS

The procedure you need to follow for changing the asymmetric master keys will differ depending on your system's hardware and coprocessor licensed internal code. Although the details of the procedures differ, they guide you through performing the same significant actions. To change the asymmetric keys, you need to:

1. Enter the master key parts into the new master key register.
2. Reencipher the PKDS under the new master key.
3. Change the asymmetric master keys and make the reenciphered PKDS the active PKDS.

The procedure for the changing the RSA master key depends on the cryptographic coprocessors on your system.

- If your system is an IBM z196 or z114 where there is a CEX3 coprocessor with September 2011 or later licensed internal code or your system is an IBM zEC12, zBC12, or later server, these are the main steps involved in performing a local RSA master key change:
  1. Enter the RSA-MK master key parts.
  2. Reencipher the PKDS under the new RSA-MK.
  3. Perform an asymmetric master key change.
- If your system is an IBM z9 or IBM z10 server or an IBM z196 or z114 where all CEX3 coprocessors have licensed internal code older than September 2011, these are the main steps involved in performing a local RSA master key change:

1. Disable PKA callable services control.
2. Enter the RSA-MK master key parts. The RSA-MK is automatically set when the final key part is loaded.
3. Reencipher the PKDS under the current RSA-MK.
4. Perform an asymmetric master key change.
5. Enable PKA callable services control.

Steps for reenciphering the PKDS and performing a local asymmetric master key change

If running in a sysplex, see Chapter 10, "Running in a Sysplex Environment," on page 171.

Note: Prior to reenciphering a PKDS, consider temporarily disallowing dynamic PKDS update services. For more information, see "Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls" on page 109.

Before beginning this procedure, you must:

1. Enter the key parts of the new master key that you want to replace the current master key into all coprocessors on your system. For information about how to do this procedure, see "Entering master key parts" on page 111. The new master key register must be full when you change the master key.
2. Create a new VSAM data set in which the reenciphered keys will be placed to create the new reenciphered PKDS. This data set must be allocated and empty and must contain the same data set attributes as the active PKDS. For more information about defining a PKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

You can use the ICSF utility panels or a utility program to reencipher the PKDSs. See Chapter 18, "Using the ICSF Utility Program CSFPUTIL," on page 353 for instructions on how to use the utility program to reencipher a disk copy of a PKDS. If you use CSFPUTIL to reencipher the PKDS, change the asymmetric master keys using procedure below starting with step 8.

To reencipher the PKDS and change the master key using the ICSF utility panels:

1. Disable the PKA callable services control on the ICSF Administrative Control Functions panel if appropriate. See "Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls" on page 109.
2. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel: "ICSF Primary Menu panel" on page 387

   The "CSFMKM10 — Key Data Set Management panel" on page 390 appears.
3. Enter option 2, PKDS MK MANAGEMENT and the PKDS Management panel appears: "CSFMKM30 — PKDS Management panel" on page 391

   When you perform a local master key change, you must first reencipher the disk copy of the PKDS under the new master key.

   Note: If your system is using multiple coprocessors, they must have the same master key. When you change the master key in one coprocessor, you should change the master key in the other coprocessors. Therefore, to reencipher a PKDS under a new master key, the new master key registers in all coprocessors must contain the same value.

4. The “CSFCMK12 — Reencipher PKDS panel” on page 389 appears. In the Input PKDS field, enter the name of the PKDS that you want to reencipher. In the Output PKDS field, enter the name of the data set in which you want to place the reenciphered keys.

   Reenciphering the disk copy of the PKDS does not affect the in-storage copy of the PKDS. On this panel, you are working with only a disk copy of the PKDS.
5. Press ENTER to reencipher the input PKDS entries and place them into the output PKDS.

   The message REENCIPHER SUCCESSFUL appears on the top right of the panel if the reencipher succeeds.
6. If you have more than one PKDS on a disk, specify the information and press ENTER as many times as you need to reencipher all of them. Reencipher all your disk copies at this time. When you have reenciphered all the disk copies of the PKDS, you are ready to change the master key.
7. Press END to return to the “CSFMKM30 — PKDS Management panel” on page 391.
8. To change the master key select option 3, CHANGE ASYM MK, on the PKDS Management panel. The Change Asymmetric Master Key Panel appears: “CSFCMK22 — Change Asymmetric Master Key panel” on page 389
9. In the New PKDS field, enter the name of the disk copy of the PKDS that you want ICSF to place in storage.

   You should have already reenciphered the disk copy of the PKDS under the new master key. The last PKDS name that you specified in the Output PKDS field on the Reencipher PKDS panel automatically appears in this field.
10. Press ENTER.

    ICSF loads the data set into storage where it becomes operational on the system. ICSF also places the new master key into the master key register so it becomes active.

    When you press ENTER, ICSF attempts to change the master key. It displays a message on the top right of the panel. The message indicates either that the master key was changed successfully or that an error occurred that prevented the successful completion of the change process. For example, if you indicate a data set that is not reenciphered under the new master key, an error message displays and the master key is not changed.
11. When performing a local change master key, remember to change the name of the PKDS in the Installation Options Data Set.

Performing a coordinated change master key

The coordinated change master key function simplifies the procedure for changing the master keys that are used by the CKDS and PKDS. Coordinated change master key may be performed on a single instance of ICSF, on a single-system sysplex, or on a multi-system sysplex.

A coordinated KDS change master key can be done by writing an application to invoke the Coordinated KDS Administration (CSFCRC) callable service. See *z/OS Cryptographic Services ICSF Application Programmer's Guide*z/OS Cryptographic Services ICSF Application Programmer's Guide for the description of the service.

Symmetric master keys

To perform a coordinated CKDS change master key, all members of the sysplex (including sysplex members that are not configured with the same active CKDS) must be at the ICSF FMID HCR7790 level or later.

Before performing a coordinated refresh, you should consider temporarily disallowing dynamic CKDS updates on all sysplex members for the CKDS you are processing. For information on disabling dynamic CKDS updates, see “Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls” on page 109.

Asymmetric master keys

To perform a coordinated PKDS change master key, all members of the sysplex (including sysplex members that are not configured with the same active CKDS) must be at the ICSF FMID HCR77A0 level or later.

Before performing a coordinated refresh, you should consider temporarily disallowing dynamic PKDS updates on all sysplex members for the PKDS you are processing. For information on disabling dynamic CKDS updates, see “Steps for enabling and disabling PKA callable services and Dynamic CKDS/PKDS access controls” on page 109

For the RSA master key, you need to use a TKE workstation to enter the new master key if any of your systems are IBM z9 or IBM z10 servers or an IBM z196 or z114 servers where all CEX3 coprocessors have licensed internal code older than September 2011. The ICSF Master Key Entry panels will automatically set new RSA master key loaded on coprocessors running on these systems. Coordinated change master key can only be performed when new master keys are loaded in the new master key register.

Usage notes

- Reenciphering a large KDS (millions of records) may cause a temporary internal suspension of KDS update requests running in parallel. If you cannot tolerate a temporary suspension in your CKDS or PKDS workload and would prefer that update requests are failed instead of suspended, you should disallow dynamic CKDS or PKDS access prior to performing the coordinated KDS change master key.
- This procedure is only for reenciphering the active KDS. It is not for reenciphering archived or backup KDS copies that are not currently active.
- If you have a combination of cryptographic coprocessors installed in a sysplex environment, the ICSF instance configured with the cryptographic coprocessor containing the highest level of licensed internal code must initiate the coordinated change master key. If the coordinated change master key is not initiated by the ICSF instance containing the highest level of licensed internal code, the operation will fail.

Steps to performing a coordinated KDS master key change

Before beginning this procedure, you must:

- Enter the key parts of the new master key that you want to replace the current master key into all coprocessors on your system. For information about how to do this procedure, see “Entering master key parts” on page 111. The new master key register must be full when you change the master key.

- Create a new VSAM data set that will be used by coordinated change master key to place the reenciphered KDS entries. This data set must be allocated and empty and must contain the same data set attributes as the active KDS you are performing the coordinated change master key on. For more information about defining a CKDS or PKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Optionally, you may:

- Create an additional VSAM data set to serve as a backup of the new, reenciphered, KDS. This data set must be allocated and empty and must contain the same data set attributes as the active KDS you are performing the coordinated change master key on.
- If you are planning to use the archive option, which is described below, determine a VSAM data set name to use for the archived KDS data set. This data set must not be allocated and must not exist on the system. For more information about defining a CKDS or PKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

To reencipher the KDS and change the master key:

1. Enter option 2, KDS MANAGEMENT, on the "ICSF Primary Menu panel" on page 387 to access the Master key set or change, KDS processing panel.
2. The "CSFMKM10 — Key Data Set Management panel" on page 390 is displayed.
   - Enter option 1 for CKDS MK Management and the CKDS Management panel appears: "CSFMKM20 — CKDS Management panel" on page 390
   - Enter option 2 for PKDS MK Management and the PKDS Management panel appears: "CSFMKM30 — PKDS Management panel" on page 391
3. Enter option 5 for COORDINATED CKDS CHANGE MK option on the CKDS or the PKDS Management menu panel and the Coordinated KDS Change master key panel appears:

```
 CSFCRC20 ----------- ICSF - Coordinated KDS change master key ------------------

To perform a coordinated KDS change master key, enter the KDS names below
and optionally select the rename option.

    KDS Type ===> CKDS

  Active KDS ===> 'CSF.CKDS'

     New KDS ===>

          Rename Active to Archived and New to Active (Y/N) ===> N

          Archived KDS ===>

          Create a backup of the reenciphered KDS (Y/N) ===> N

          Backup KDS ===>

Press ENTER to perform a coordinated KDS change master key.
Press END to exit to the previous menu.
```

   In this example, CKDS was selected to perform the coordinated change master key. The KDS type is displayed in the **KDS Type** field. The active KDS is displayed in the **Active KDS** field.

a. Enter the name of the new KDS in the **New KDS** field. This must be an empty and allocated VSAM data set containing the same data set attributes as the active KDS. The reenciphered keys will be placed into this new data set to create the new KDS.

b. Decide if you want to have the new KDS renamed to the match the name of the current active KDS. Having the new KDS renamed to match the name of the current active KDS simplifies KDS administration, because you will not need to update the ICSF Options Data Set with the name of the new data set after the coordinated change master key completes.

- If you would like the have the new KDS renamed to match the name of the current active KDS:
  1) Type Y in the **Rename Active to Archived and the New to Active (Y / N)** field.
  2) Enter the name under which the currently active KDS will be archived in the **Archived KDS** field. This must be a VSAM data set name that is not allocated and does not exist on the system.
- If you do not want to have the new KDS renamed to match the name of the current active KDS, type N in the **Rename Active to Archived and the New to Active (Y / N)** field. Remember to change the name of the KDS in the Installation Options Data Set as described in the *z/OS Cryptographic Services ICSF System Programmer's Guide*. The KDS name must be changed in each cluster member's Installation Options Data Set after the coordinated KDS change master key function completes successfully. If the Installation Options Data Set is updated with a new KDS name and the coordinated change master key function fails, ICSF might be configured with an invalid KDS the next time it is restarted.

c. Decide if you want to also create a backup copy of the newly enciphered KDS. This is an empty and allocated VSAM data set containing the same data set attributes as the active KDS. The reenciphered keys will be placed into this data set to create the backup KDS.

4. Press ENTER to begin the coordinated change master key.
5. A confirmation panel will be displayed, prompting you to verify that you want to continue with the coordinated change master key. Verify that the information on this confirmation panel is correct. If it is, type Y in the confirmation field provided and press ENTER.

   The coordinated change master key function will be executed. This function will verify that all ICSF instances sharing the same active KDS are configured with the same New Master Key registers values. Additionally, it will verify that the KDS names specified for input are valid and are compatible with each other. The disk copy of the active KDS will be reenciphered under the new master keys to create the new KDS on disk and will create an in-storage copy of that new KDS. In a sysplex environment, the in-storage copy of the new KDS will be created for all ICSF instances that share the KDS.
6. Verify the dialog results and correct any indicated failures or unexpected results.

Recovering from a coordinated administration failure

This information describes how to use ICSF diagnostic information to recover from a coordinated administration failure.

The coordinated administration functions performs multiple steps to validate the environment, including verifying master key registers across the sysplex cluster and validating KDSs involved in the operation. If the environment is verified and meets criteria for the operation, then the initiating system of the coordinated administration function will attempt to coordinate the function across all members of the sysplex cluster (all ICSF instances sharing the same active KDS).

Coordinated change master key and coordinated refresh messages

The coordinated refresh and coordinated change master key dialogs result in one or more dialog messages indicating the success or failure of the operation. In the case of a failure, there should be enough information in the dialog message to identify the problem. If there is not enough information in the dialog, you must use the ICSF job log to further identify the problem.

During coordinated change master key and coordinated refresh, a sequence of messages are written to the ICSF job log. CSFM622I messages are written to provide status for internal steps taken by the function. For example, one of the very first steps for a coordinated change master key operation is to make a copy of the in-storage KDS that will be used for the subsequent reencipher step. When this copy is made, the following CSFM622I message is written to the ICSF joblog.

```
CSFM622I COORDINATED CHANGE-MK PROGRESS: NEW IN-STORAGE KDS CONSTRUCTED.
```

If a failure occurs during coordinated change master key or coordinated refresh, failure messages are written to the ICSF job log that provide diagnostic information for determining the cause of the problem. Depending on how far the function is into processing, steps may be required to back out from the overall operation. CSFM622I messages are also used to provide status for back out steps. Additionally, all failure cases will end with the following CSFM616I message to provide further diagnostic information.

```
CSFM616I COORDINATED operation FAILED, RC=return-code RS=reason-code
         SUPRC=supplemental-return-code SUPRS=supplemental-reason-code
         FLAGS=flags.
```

An explanation of the return code and reason code provided in the CSFM616I message can be found in the "Return and Reason Codes" section of the *z/OS Cryptographic Services ICSF Application Programmer's Guide*. The rest of the information in this message is IBM internal diagnostic information.

The sequence of messages written to the ICSF job log during a coordinated change master key and coordinated refresh should indicate how far along the function progressed, and, if a failure occurred, should include enough diagnostic information to determine the cause of the problem. Use the CSFM622I messages to determine how far along the function progressed before the failure. Then use the failure messages to determine why the problem occurred.

New master key register mismatch

For a coordinated change master key, all sysplex cluster members must have their new master key registers pre-loaded with the same master key values. For coordinated CKDS change master key, one of either the DES or AES new master key registers must be pre-loaded, or both may be pre-loaded on all sysplex cluster members. For coordinated PKDS change master key, one of either the RSA or ECC new master key registers must be pre-loaded, or both may be pre-loaded on all sysplex cluster members. For coordinated TKDS change master key, the P11 or RCS new master key register must be pre-loaded and committed on all sysplex cluster members.

If a sysplex cluster member's new master key registers do not match the initiator's new master key registers, the following error message is displayed on the dialog. In this example, the coordinated CKDS change master key failed because the AES new master key register did not match on one of the sysplex cluster members systems.

```
THE AES NEW MASTER KEY REGISTERS ARE NOT CONSISTENT ACROSS ALL COPROCESSORS FOR
THE SYSPLEX SYSTEMS PARTICIPATING IN THIS OPERATION. THEY MUST BE THE SAME. THIS
ERROR WAS DETECTED ON A SYSTEM OTHER THAN THIS ONE.
```

In addition, the following message is written to the ICSF job log.

```
CSFM615I COORDINATED CHANGE-MK FAILED. NEW MASTER KEYS INCORRECT ON sysname.
         RC = return-code, RSN = reason-code.
```

To resolve this problem, the security administrator should compare all sysplex cluster members' new master key registers to ensure that they match the initiator's exactly. If a sysplex cluster member's new master key registers do not match, the security administrator should re-load them or clear them to match the values on the initiating system.

Additional information about the failure can be determined by looking up the return and reason codes in the “Return and Reason Codes” section of the *z/OS Cryptographic Services ICSF Application Programmer’s Guide*.

Cataloged failures

If any of these data sets are not cataloged, one of the following dialog messages will be displayed:

```
ICSF COULD NOT SUFFICIENTLY RESOLVE SYSTEM CATALOG INFORMATION FOR ONE OF THE
CURRENTLY ACTIVE OR NEW DATA SETS. REFER TO THE ICSF JOBLOG(S) FOR ADDITIONAL
INFO. IBM DIAGNOSTIC INFORMATION: RROPRC=0000000C RROPRSN=00000C2C SUPPRC=00000000
SUPPRSN=00000000 FLAGS=02800000
```

```
ICSF SUFFERED AN UNEXPECTED I/O ERROR REFERENCING OR UPDATING ONE OF THE ACTIVE,
NEW OR BACKUP DATA SETS. REFER TO THE ICSF JOBLOG(S) FOR ADDITIONAL INFO. IBM
DIAGNOSTIC INFORMATION: RROPRC=0000000C RROPRSN=00002740 SUPPRC=0000000C
SUPPRSN=00001790 FLAGS=41800000
```

In addition, a CSFM619I message, a CSFM623I message, or both messages will be written to the ICSF job log.

To correct this problem, make sure the necessary data sets are cataloged and retry the function.

Mainline processing failure

If a coordinated change master key or coordinated refresh operation fails during one of its internal mainline processing steps, a dialog message will be displayed indicating the problem and a CSFM620I message will be written to the ICSF job log.

For example, when using the rename option, if the active KDS cannot be renamed to the archive data set name, the following dialog message will be displayed:

```
ICSF WAS UNABLE TO SUCCESSFULLY RENAME ONE OF THE ACTIVE DATA SET TO THE
ARCHIVE NAME, OR THE NEW DATA SET TO THE ACTIVE NAME. REFER TO THE ICSF
JOBLOG(S) FOR ADDITIONAL INFO. IBM DIAGNOSTIC INFORMATION: RROPRC=0000000C
RROPRSN=00000C3E SUPPRC=0000000C SUPPRSN=00000C3E FLAGS=41800000
```

In addition, the following message will be written to the ICSF job log:

```
CSFM620I COORDINATED CHANGE-MK MAINLINE PROCESSING FAILED BECAUSE THE ACTIVE DATA
         SET CANNOT BE RENAMED TO THE ARCHIVE NAME.
```

To correct this problem the security administrator or system programmer should determine if there is a conflict with the archive data set name that caused the failure. The CSFM620I is also used for other internal mainline processing failures, such as if a problem occurs trying to load or process the target or backup data sets. For either case, the CSFM620I message should provide enough information for the security administrator or system programmer to further investigate the problem.

Backout processing failure

If a failure occurs during mainline processing of a coordinated change master key or coordinated refresh, backout processing will attempt to undo any steps that have already completed in the operation.

A CSFM620I message will be written to the ICSF job log to indicate the mainline processing failure. Additionally backout processing messages will be written to the ICSF job log indicating the status of the backout.

If backout processing fails, a dialog message will indicate the problem. For example:

```
ICSF PROCESSING SUFFERED AN UNRECOVERABLE ERROR AND WAS FORCED TO SHUT
DOWN ACROSS PARTICIPATING SYSPLEX SYSTEMS TO AVOID A POTENTIAL INCONSISTENT
ENVIRONMENT. REFER TO THE ICSF JOBLOG(S) FOR ADDITIONAL INFO. IBM DIAGNOSTIC
INFORMATION: RROPRC=0000000C RROPRSN=0000C3E SUPPRC=0000000C SUPPRSN=00000C42
FLAGS=C5800000
```

A series of CSFM622I messages will be written to the ICSF joblog to track the status of the back out steps. If there is a failure during backout processing, a CSFM621I message will be written to the ICSF job log indicating the failure during backout processing.

When a failure in backout processing occurs, use the overall sequence of CSFM620I, CSFM621I, and CSFM622I messages to determine which step the function failed on, and which step failed during backout processing. For this situation, it is likely other messages listed in this section are also written to the ICSF job log to help determine the root cause of the problem.

Set master key failure

If there was a problem setting the master key on either the initiating system or a target system of a coordinated change master key, a dialog message will indicate the failure and a CSFM625I message will be written to that system's ICSF job log.

For example, if the step for setting the AES master key fails, the following dialog message will be displayed:

```
A SET-MASTER-KEY ACTION FAILED ON THIS SYSTEM. REFER TO THE ICSF JOBLOG(S) FOR
ADDITIONAL INFO.
```

The following message will be written to the ICSF job log for this failure.

```
CSFM625I SET AES MASTER KEY FAILED FOR COPROCESSOR SERIAL NUMBER 93X06032.
```

If this failure occurs on the initiating system, the entire change master key processes will be canceled and the target systems will not be affected by the operation. Check the status of the coprocessor with the serial number identified in the message to determine if it requires maintenance.

If this failure occurs on a target system, the initiating system and other target systems may have successfully changed their master key. If the initiating system has set the master key and completed the coordinated change master key, the active KDS is now enciphered under the new master key. Check the status of the coprocessor with the serial number identified in the message to determine if it requires maintenance. After the coprocessor's status is resolved, the target system must manually set the new master key value to remain in synch with the active KDS. For CKDS and PKDS, follow the steps in “Reentering master keys when they have been cleared” on page 126 (Any CCA cryptographic coprocessor). For TKDS, follow the steps in “Re-entering Master Keys after they have been cleared” on page 165.

Back-level ICSF releases in the sysplex

The coordinated CKDS change master key and coordinated CKDS refresh functions are only available if all ICSF instances in the CKDS sysplex group are running ICSF FMID HCR7790 or later. If an ICSF instance at a level lower than ICSF FMID HCR7790 joins the sysplex group, a CSFM631I message (indicating all down-level systems) will be written to the ICSF job log and the operation will fail.

The coordinated PKDS change master key and coordinated PKDS refresh functions are only available if all ICSF instances in the PKDS sysplex groups are running ICSF FMID HCR77A0 or later. If an ICSF instance at a level lower than ICSF FMID HCR77A0 joins the sysplex group, a CSFM631I message (indicating all down-level systems) will be written to the ICSF job log and the operation will fail.

The coordinated TKDS change master key function is only available if all ICSF instances in the TKDS sysplex groups are running ICSF FMID HCR77A0 or later. If an ICSF instance at a level lower than ICSF FMID HCR77A0 joins the sysplex group, a CSFM631I message (indicating all down-level systems) will be written to the ICSF job log and the operation will fail.

To resolve this problem, all down-level systems must either be removed from the KDS sysplex group or upgraded to the required level or higher. If this is not possible, the coordinated change master key and coordinated refresh functions cannot be used. The local CKDS change master key, local CKDS refresh, local PKDS change master key, and local PKDS refresh can be used with ICSF instances running at supported FMID levels. The TKDS does not have a local change master key, local refresh, or coordinated refresh function.

Rename failures

If there is a failure during the optional rename step of coordinated change master key or coordinated refresh, CSFM629I and CSFM630I messages will be written to the ICSF job log to indicate the reason for the failure.

The rename function uses the IDCAMS processor to perform the actual VSAM data set rename. CSFM629I messages are used to route IDCAMS processor messages to the ICSF job log when the IDCAMS processor fails to perform the rename. The CSFM629I messages contain the reason from the IDCAMS failure. These messages are followed by a CSFM630I message that indicates which data set name failed to be renamed to which new name.

ICSF KDS data sets are KDS VSAM data sets. They consist of three parts: a cluster name, an index name, and a data name. For example, if you use the sample JCL provided in the “Steps to create the CKDS” section of the *z/OS Cryptographic Services ICSF System Programmer's Guide*, the cluster name, data name, and index name will be the following in order.

CSF.CSFCKDS
CSF.CSFCKDS.DATA
CSF.CSFCKDS.INDEX

When the rename option is selected, all three parts of the active KDS will be renamed to the archive name, and all three parts of the target KDS will be renamed to the active name. When renaming the data and index portions of a KDS VSAM data set, the suffix format of the original data set is maintained. For example, if the preceding data set names are used for the active CKDS, and the archive data set name is specified as CSF.CSFCKDS.ARC, the three portions of the active CKDS will be renamed to:

CSF.CSFCKDS.ARC
CSF.CSFCKDS.ARC.DATA
CSF.CSFCKDS.ARC.INDEX

In the case of a failure during rename processing, the coordinated function will attempt to back out and rename the data sets back to their original names. If the back out fails, you may end up with a partially renamed data set. This can be easily corrected by performing an IDCAMS ALTER from JCL.

Whenever a rename failure occurs, scan the ICSF job log of the initiating system for CSFM629I and CSFM630I messages. These messages will indicate which data set part failed during rename and if backout processing was able to rename the data set back to its original name. If backout processing was able to rename back to the original name, check the catalog for the data set name that failed to be used for rename. Most likely you have a conflict with the archive data set name and need to either rename existing data sets in your catalog or choose a different archive name.

If backout processing failed to rename your data set back to the original name, use ISPF to confirm that the data set parts match up with what is reported in the CSFM629I and CSFM630I messages.

For example, if during a coordinated CKDS change master key operation, the active CKDS cluster name is successfully renamed to the archive name, but the data portion fails to be renamed, backout processing begins. If backout processing fails to rename the data set back to the original active CKDS name, ICSF will shut down all instances in the sysplex CKDS cluster because the active CKDS name is only half renamed. In this scenario, the following set of messages may be reported in the ICSF job log.

```
CSFM618I CKDS DATA SET CSF.CSFCKDS RENAMED TO CSF.CSFCKDS.ARC
CSFM629I IDCAMS  SYSTEM SERVICES                                 TIME: 13:35:12       06/07/11
CSFM629I
CSFM629I  ALTER CSF.CSFCKDS.DATA                     -
CSFM629I  NEWNAME(CSF.CSFCKDS.ARC.DATA             )
CSFM629I IDC3013I DUPLICATE DATA SET NAME
CSFM629I IDC3009I ** VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGG0CLE6-8
CSFM629I IDC0532I **ENTRY CSF.CSFCKDS.DATA NOT ALTERED
CSFM629I IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8
CSFM629I
CSFM629I IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 8
CSFM630I CKDS RENAME FAILED: CSF.CSFCKDS.DATA TO CSF.CSFCKDS.ARC.DATA
CSFM629I IDCAMS  SYSTEM SERVICES                                 TIME: 13:35:18       06/07/11
```

```
CSFM629I
CSFM629I  ALTER CSF.CSFCKDS.ARC                     -
CSFM629I  NEWNAME(CSF.CSFCKDS                        )
CSFM629I IDC3013I DUPLICATE DATA SET NAME
CSFM629I IDC3009I ** VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGG0CLE6-8
CSFM629I IDC0532I **ENTRY CSF.CSFCKDS.ARC NOT ALTERED
CSFM629I IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8
CSFM629I
CSFM629I IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 8
CSFM630I CKDS RENAME FAILED: CSF.CSFCKDS.ARC TO CSF.CSFCKDS
CSFM620I COORDINATED CHANGE-MK MAINLINE PROCESSING FAILED BECAUSE THE ACTIVE DATA SET
         CANNOT BE RENAMED TO THE ARCHIVE NAME.
CSFM622I COORDINATED CHANGE-MK PROGRESS: BACKOUT IS BEING DRIVEN BY MAINLINE.
CSFM629I IDCAMS  SYSTEM SERVICES                                    TIME: 13:35:24       06/07/11
CSFM629I
CSFM629I  ALTER CSF.CSFCKDS.ARC                     -
CSFM629I  NEWNAME(CSF.CSFCKDS                        )
CSFM629I IDC3013I DUPLICATE DATA SET NAME
CSFM629I IDC3009I ** VSAM CATALOG RETURN CODE IS 8 - REASON CODE IS IGG0CLE6-8
CSFM629I IDC0532I **ENTRY CSF.CSFCKDS.ARC NOT ALTERED
CSFM629I IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8
CSFM629I
CSFM629I IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 8
CSFM630I CKDS RENAME FAILED: CSF.CSFCKDS.ARC TO CSF.CSFCKDS
CSFM621I COORDINATED CHANGE-MK BACK OUT PROCESSING FAILED BECAUSE THE ARCHIVE DATA SET
         CANNOT BE RENAMED TO THE ACTIVE NAME.
CSFM621I COORDINATED CHANGE-MK BACK OUT PROCESSING FAILED BECAUSE ICSF IS UNABLE TO
         RELIABLY RESTORE THE ORIGINAL ACTIVE KDS.
CSFM622I COORDINATED CHANGE-MK PROGRESS: CANCELING CORE WORK.
CSFM616I COORDINATED CHANGE-MK FAILED, RC=0000000C RS= 00000C3E SUPRC= 0000000C SUPRS=
         00000C42 FLAGS= C5800000.
CSFU006I CHANGE-MK FEEDBACK: CC=0000000C RSN=00000C3E SUPPRC=0000000C SUPPRSN=00000C42
         FLAGS=C5800000.
CSFM308I MEMBER XXX REPORTED REMOVED FROM SYSPLEX GROUP SYSICSF.
CSFM308I MEMBER XXX REPORTED REMOVED FROM SYSPLEX GROUP SYSICSF.
CSFM401I CRYPTOGRAPHY - SERVICES ARE NO LONGER AVAILABLE.
```

This sequence of messages indicates that the active CKDS name of CSF.CSFCKDS was renamed to CSF.CSFCKDS.ARC. Then, the CSF.CSFCKDS.DATA data portion of the active CKDS failed to be renamed to CSF.CSFCKDS.DATA.ARC because another data set in the catalog was already using this name. At this point, the coordinated CKDS change master key function tried to back out and rename the cluster portion of the CKDS from CSF.CSFCKDS.ARC back to its original name of CSF.CSFCKDS. However the renaming failed because another data set with name CSF.CSFCKDS now exists in the catalog. The result is a half renamed active CKDS which causes ICSF to shut down across the CKDS sysplex cluster.

The first step to resolving this problem is to confirm in ISPF that the following data set names reported in the messages above do exist:

CSF.CSFCKDS.ARC
CSF.CSFCKDS.DATA
CSF.CSFCKDS.INDEX

Once this is confirmed, the next step is to rename the cluster name back to the original name manually by calling IDCAMS ALTER from JCL. Before doing that, the messages above indicate that back out processing already failed to rename the cluster name back because another data set is now using that name. The data set that has taken that name should be renamed to a different name as this name is needed to restore the active CKDS.

Once the cluster name conflict has been resolved, issue IDCAMS ALTER from JCL to rename the CSF.CSFCKDS.ARC cluster name back to the original active CKDS name of CSF.CSFCKDS.

For example:

```
//DEFINE   EXEC  PGM=IDCAMS,REGION=4M
//SYSPRINT  DD   SYSOUT=*
//SYSIN  DD  *
     ALTER  CSF.CSFCKDS.ARC   -
            NEWNAME(CSF.CSFCKDS)
/*
```

ICSF may be restarted on all instances that were previously taken down. Processing should resume as normal and the coordinated CKDS change master key with rename option may be issued again with an archive data set name that does not have a conflict in the catalog.

Adding cryptographic coprocessors after initialization

You may need to initialize coprocessors after system initialization.

Note: If you used the Pass Phrase Initialization (PPINIT) utility to initialize your system, use the utility to added coprocessors. See "Steps for adding a CCA coprocessor after first time Pass Phrase Initialization" on page 100.

This is the procedure:

1. Load the new master key register with the current master key values using the ICSF Master Key Entry panel (see "Steps for entering the first master key part" on page 117) or the TKE workstation.
2. When the new master key register is loaded, SET the master key. Enter option 2, KDS MANAGEMENT, on the ICSF Primary Menu panel to access the Key Data Set Management panel: "ICSF Primary Menu panel" on page 387

   The "CSFMKM10 — Key Data Set Management panel" on page 390 appears.
3. Choose option 4, SET MK.

   The master keys will be set if the verification pattern of the new master key register matches the MKVP in the key data sets. All new master key registers that match will be set.

Clearing master keys

For security reasons, your installation may need to clear the master keys. This may be required, for example, prior to turning the processor hardware over for maintenance.

If you have a TKE workstation, you can use it to zeroize all domains that have keys loaded. Refer to *z/OS Cryptographic Services ICSF TKE Workstation User's Guide* for more information.

If you do not have a TKE workstation, you might want to consider nullifying the master keys. To do this you would need to enter new master keys for the master key you have loaded, reencipher a dummy CKDS and PKDS, and change the master keys. You would need to perform this operation twice to ensure that the master keys are cleared from the old master key register.

You can also use the zeroize function on the Support Element panel. Besides clearing the master keys, this also clears all domains and installation data.

Chapter 8. Managing PKCS #11 master keys

This topic describes how to manage master keys for Enterprise PKCS #11 (EP11) coprocessors and regional cryptographic servers.

- For Enterprise PKCS #11, a master key is used to protect all secure PKCS #11 keys that are active on your system. This master key is known as the P11 master key or P11-MK.
- For regional cryptographic services, an RCS-MK master key is used to protect RCS keys that are active on your system. The RCS-MK must be loaded using your RCS vendor utility supplied with the regional cryptographic server.

Because master key protection is essential to the security of the other keys, master keys are only stored within the hardware of the cryptographic coprocessor or regional cryptographic server. This nonvolatile key storage area is unaffected by system power outages because it has a battery backup or is stored to flash memory. The values of the master keys never appear in the clear outside the hardware of the cryptographic coprocessor or regional cryptographic server.

A TKE workstation is required to load key parts for the P11 master key. Unlike the CCA coprocessors, this cannot be done using ICSF alone. Be prepared to switch between your TKE workstation and your ICSF host session. For more information, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Important: The master keys are loaded into the new master key register. The utilities described in this topic require the master key to be in the new master key register. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. This option must not be used for a coprocessor or domain that is visible to ICSF. Setting the master keys from the TKE workstation may result in the loss of the cryptographic function of that coprocessor or domain.

New master key automatically set when ICSF started

ICSF automatically sets new master key registers when ICSF is started if the current master key does not match the MKVP in the TKDS. The check of the new master key registers is made every time ICSF is started.

The following conditions must be true:

- The new master key register must be full (full committed).
- The master key verification pattern (MKVP) of the new master key register must match the MKVP in the header of the TKDS.

Entering master key parts using the TKE workstation

P11 master key parts are loaded using smart cards only. You may enter up to 20 master keys parts.

Note:

1. If your ICSF instance (single system or LPAR image) is using multiple EP11 coprocessors, they must have the same master key. Therefore, the new master key registers in all EP11 coprocessors must contain the same value.

2. If you are reentering master keys after they have been cleared, use the same master key part values as when you originally entered the keys.

First time use of Enterprise PKCS #11 keys

ICSF PKCS #11 services may be utilized for clear key operations both with and without a TKDS. To use secure PKCS #11 keys, a TKDS is required. The first time you intend to use secure PKCS #11 services, you must load a P11-MK and initialize a new, empty TKDS or update your existing, clear key only TKDS. For information on creating an empty TKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*. When you initialize/update the TKDS, ICSF creates a header record for the TKDS, installs the required P11-MK key pattern in the TKDS, and sets the master key. All secure PKCS #11 keys stored in the TKDS are enciphered under the P11-MK. (Note, the TKDS may contain both clear and secure keys, if desired.) After the master key has been set, you can generate or enter any keys you need to perform secure PKCS #11 cryptographic functions.

To begin, load the P11 new master key register using the TKE workstation. If you intend to have multiple instances of ICSF sharing the same active TKDS in a sysplex environment, you can define an EP11 domain group on the TKE workstation to load the same P11-MK in all domains used by all ICSF instances that will share the same active TKDS. After loading the new P11 master key, commit the new P11 master key using the TKE workstation.

After the P11 new master key register has been loaded and committed, the TKDS must be initialized. TKDS initialization is only required the first time the P11 new master key register is loaded. When sharing the TKDS in a sysplex environment, TKDS initialization should be performed on each ICSF instance sharing the TKDS. Optionally, after you initialize a TKDS on one ICSF instance, you can then share it with other ICSF instances that are configured with the same P11 master key value, by simply starting up or restarting the other ICSF instances.

Initialize or update the TKDS

| If running in a sysplex, see Chapter 10, “Running in a Sysplex Environment,” on
| page 171.

At this point, the new P11 master key register on each EP11 coprocessor available to this ICSF instance must be in the FULL COMMITTED state. If running in a Sysplex, the new P11 master key register in each domain sharing the TKDS should also be FULL COMMITTED with the same master key parts. You must now initialize a new (or update the existing) TKDS, thus activating the P11-MK.

1. From the ICSF Primary Menu, select option 2, KDS MANAGEMENT.

```
HCR77C0 ------------- Integrated Cryptographic Service Facility ---------

OPTION ===>

Enter the number of the desired option.

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

Figure 37. Selecting KDS MANAGEMENT on the ICSF primary menu panel

2. The Key Data Set Management panel appears. Select option 3, TKDS MK MANAGEMENT.

```
CSFMKM10 ---------------- ICSF – Key Data Set Management ----------------
OPTION ===>

Enter the number of the desired option.

  1  CKDS MK MANAGEMENT - Perform Cryptographic Key Data Set (CKDS)
                          functions including master key management
  2  PKDS MK MANAGEMENT - Perform Public Key Data Set (PKDS)
                          functions including master key management
  3  TKDS MK MANAGEMENT - Perform PKCS #11 Token Data Set (TKDS)
                          functions including master key management
  4  SET MK             - Set master key
```

Figure 38. Selecting TKDS MK MANAGEMENT on the Key Data Set Management panel

3. The TKDS Master Key Management panel appears. Select option 1, INIT/UPDATE TKDS. This causes ICSF to update the header record of the active TKDS and set the master key. No additional sub-panels are shown.

```
CSFMKM40 ------------------- ICSF - TKDS Master Key Management ----------------------
OPTION ===>

Enter the number of the desired option.

  1  INIT/UPDATE TKDS - Initialize the active TKDS or update the header of
                        the active TKDS
  2  COORDINATED TKDS CHANGE MK - Perform a coordinated TKDS master key change
  3  COORDINATED TKDS CONVERSION - Convert the TKDS to use KDSR record format
```

Figure 39. Selecting INIT/UPDATE TKDS on the Key Data Set Management panel

4. When ICSF completes, the message INITIALIZATION COMPLETE appears. If you did not enter a master key into the new master key register previously, the message NMK REGISTER NOT FULL COMMITTED appears and the initialization process ends.

Note: If any part of the option 1 fails, you may need to reload the new master key register before starting over.

After you complete the entire process, the P11-MK is activated for the ICSF host system that you initiated this from. You may now start using secure PKCS #11 services. If running in a Sysplex, all instances of ICSF on other systems sharing the TKDS must perform TKDS initialization as well or must be restarted before they can start using secure PKCS #11 services.

Changing the Master Key

If running in a sysplex, see Chapter 10, "Running in a Sysplex Environment," on page 171.

For security reasons your installation should change the master keys periodically. In addition, if the master keys have been cleared, you may also want to change the master keys after you reenter the cleared master keys.

Note for P11-MK:

1. If running in a Sysplex, all ICSF instances sharing the TKDS must be at ICSF FMID HCR77A0 or higher in order to change the P11-MK, even if these systems are not using secure PKCS #11 services.
2. If your ICSF instance is using multiple coprocessors, they must have the same master key. When you change the master key in one coprocessor, you should change the master key in the other coprocessors. Therefore, to reencipher a TKDS under a new master key, the new master key registers in all Enterprise PKCS #11 coprocessors must contain the same value.
3. Changing the P11-MK can only be performed by the coordinated change master key function. All ICSF instances sharing the TKDS will have their P11 master keys changed during this process. The P11 new master key registers in each domain for each coprocessor sharing the TKDS must be loaded from TKE with the same value.

Note for RCS-MK:

1. If running in a sysplex, all ICSF instances sharing the TKDS must be running at the ICSF FMID HCR77B1 APAR OA49069 level or higher in order to change the RCS-MK, even if these systems are not using regional cryptographic services.
2. If your ICSF instance is using multiple RCS devices, they must be at the RCS generation 2 level or later and have the same master key. When you change the master key in one device, you should change the master key in the other devices. Therefore, to reencipher a TKDS under a new master key, the new master key registers in all regional cryptographic servers must contain the same value.
3. Changing the RCS-MK can only be performed by the coordinated change master key function. All ICSF instances sharing the TKDS will have their RCS master keys changed during this process. The RCS new master key register for each active RCS device sharing the TKDS must be loaded with the same value from your RCS vendor utility.

To begin, load the new P11 master key using the TKE workstation and P11 master key parts stored on smart cards or load the new RCS master key using your RCS vendor utility. Then commit the new P11 or RCS master key using the TKE workstation or your RCS vendor utility. For more information, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Create a new VSAM data set in which the reenciphered keys will be placed when creating the new reenciphered TKDS. This data set must be allocated and empty, and must contain the same data set attributes as the active TKDS. For more information about defining a TKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Optionally, you may:

- Create an additional VSAM data set to serve as a backup of the new, reenciphered, KDS. This data set must be allocated and empty, and must contain the same data set attributes as the active KDS you are performing the coordinated change master key on.
- If you are planning to use the archive option, determine a VSAM data set name to use for the archived TKDS data set. This data set must not be allocated and must not exist on the system. For more information about defining a TKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

1. From the ICSF Primary Menu, select option 2, KDS MANAGEMENT.

```
HCR77C0 ------------- Integrated Cryptographic Service Facility ---------

OPTION ===>

Enter the number of the desired option.

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

Figure 40. Selecting KDS Management on the ICSF primary menu panel

2. The Key Data Set Management panel appears. Select option 3, TKDS MK MANAGEMENT.

```
CSFMKM10 ---------------- ICSF – Key Data Set Management ----------------
OPTION ===>

Enter the number of the desired option.

  1  CKDS MK MANAGEMENT - Perform Cryptographic Key Data Set (CKDS)
                          functions including master key management
  2  PKDS MK MANAGEMENT - Perform Public Key Data Set (PKDS)
                          functions including master key management
  3  TKDS MK MANAGEMENT - Perform PKCS #11 Token Data Set (TKDS)
                          functions including master key management
  4  SET MK             - Set master key
```

Figure 41. Selecting TKDS MK MANAGEMENT on the Key Data Set Management panel

3. The TKE Key Data Set Management panel appears. Select option 2, COORDINATED TKDS CHANGE MK.

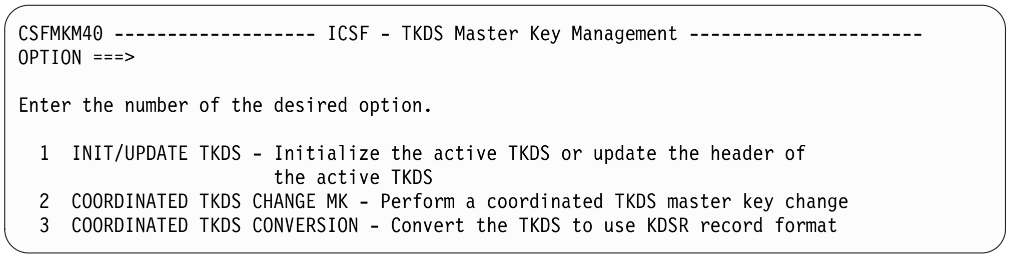

```
CSFMKM40 ------------------- ICSF - TKDS Master Key Management ----------------------
OPTION ===>

Enter the number of the desired option.

  1  INIT/UPDATE TKDS - Initialize the active TKDS or update the header of
                       the active TKDS
  2  COORDINATED TKDS CHANGE MK - Perform a coordinated TKDS master key change
  3  COORDINATED TKDS CONVERSION - Convert the TKDS to use KDSR record format
```

Figure 42. Selecting COORDINATED TKDS CHANGE MK on the Key Data Set Management panel

4. The Coordinated KDS change master key panel is displayed.

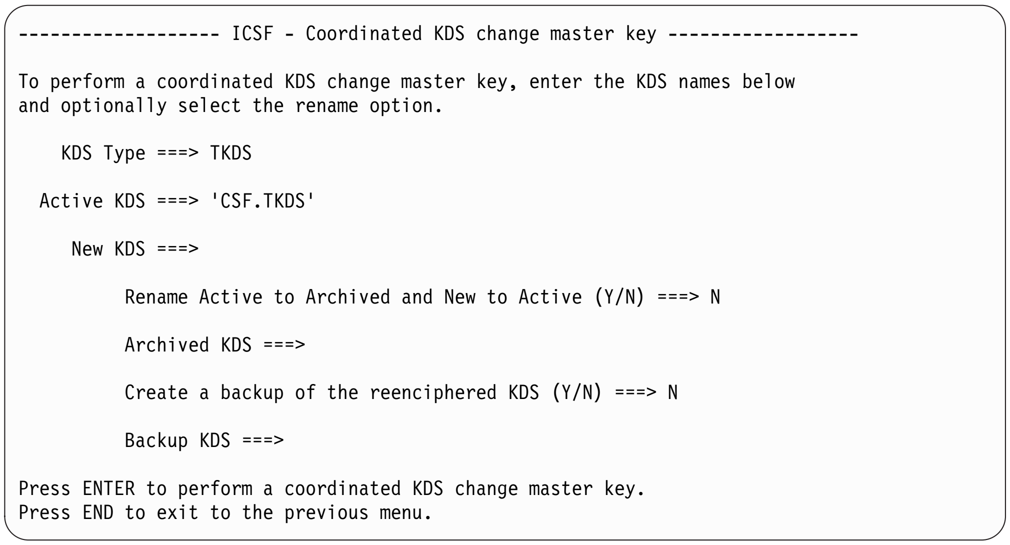

```
------------------- ICSF - Coordinated KDS change master key ------------------

To perform a coordinated KDS change master key, enter the KDS names below
and optionally select the rename option.

    KDS Type ===> TKDS

  Active KDS ===> 'CSF.TKDS'

     New KDS ===>

          Rename Active to Archived and New to Active (Y/N) ===> N

          Archived KDS ===>

          Create a backup of the reenciphered KDS (Y/N) ===> N

          Backup KDS ===>

Press ENTER to perform a coordinated KDS change master key.
Press END to exit to the previous menu.
```

Figure 43. Coordinated KDS change master key panel

The KDS type (TKDS) is displayed in the KDS Type field. The active TKDS is displayed in the Active KDS field.

a. Enter the name of the new TKDS in the New KDS field. This is an empty and allocated VSAM data set containing the same data set attributes as the active TKDS. The reenciphered keys will be placed into this new data set to create the new TKDS.
b. Decide if you want to have the new TKDS renamed to the match the name of the current active TKDS. Having the new TKDS renamed to match the name of the current active TKDS simplifies TKDS administration, because you will not need to update the ICSF Options Data Set with the name of the new data set after the TKDS is reenciphered.

If you would like the have the new TKDS renamed to match the name of the current active TKDS:

a. Type Y in the Rename Active to Archived and the New to Active (Y / N) field.
b. Enter the name under which the currently active TKDS will be archived in the Archived KDS field. This must be a VSAM data set name that is not allocated and does not exist on the system.

   If you do not want to have the new TKDS renamed to match the name of the current active TKDS, type N in the Rename Active to Archived and the New to Active (Y / N) field. Remember to change the name of the TKDS in the Installation Options Data Set as described in the *z/OS Cryptographic Services ICSF System Programmer's Guide*.

   c. Decide if you want to also create a backup copy of the newly enciphered TKDS. This is an empty and allocated VSAM data set containing the same data set attributes as the active TKDS. The reenciphered keys will be placed into this data set to create the backup TKDS.

5. Press ENTER to begin the coordinated change master key. This will reencipher the disk copy of the active TKDS under the new master keys to create the new TKDS on disk. This will also create an in-storage copy of that new TKDS and activate (set) the new P11, RCS, or both P11 and RCS master keys on the Enterprise PKCS #11 coprocessors, regional cryptographic servers, or Enterprise PKCS #11 coprocessors and regional cryptographic servers.
6. A confirmation panel will be displayed, prompting you to verify that you want to continue with the coordinated change master key. Verify that the information on this confirmation panel is correct. If it is, type Y in the confirmation field provided and press ENTER.

   The coordinated change master key function will be executed. When ICSF completes, the message CHANGE MK SUCCESSFUL appears. In the case of a failure, see "Recovering from a coordinated administration failure" on page 150 for assistance in resolving failures.

   Note:

   a. In a sysplex environment, the in-storage copy of the new TKDS will be created (and new P11 master key activated) for all ICSF instances that share the same active TKDS.
   b. For more information about the coordinated change master key function, see "Coordinated change master key and coordinated refresh utilities" on page 171.

Re-entering Master Keys after they have been cleared

In these situations, the Enterprise PKCS #11 coprocessor clears the master key registers so that the master key values are not disclosed:

- If the coprocessor detects tampering (the intrusion latch is tripped), ALL installation data is cleared: master keys and (optionally) all installed administrators.
- If the coprocessor detects tampering (the secure boundary of the card is compromised), it self-destructs and can no longer be used.
- If you issue a command from the TKE workstation to zeroize a domain or the entire cryptographic feature.
- If you issue a command from the Support Element panel to zeroize the entire cryptographic feature.

For situations when the regional cryptographic server would clear the RCS master key, consult your RCS vendor documentation.

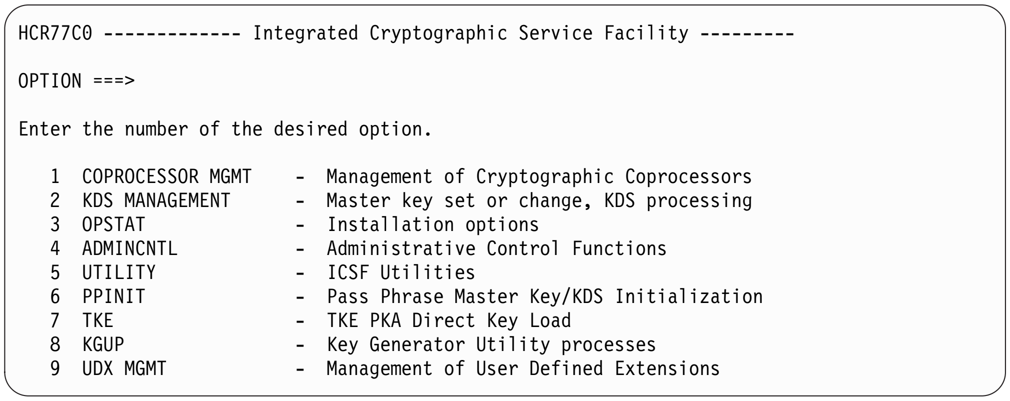
Although the values of the master keys are cleared, the secure keys in the TKDS are still enciphered under the cleared P11 or RCS master key. Therefore, to recover these secure keys, you must reenter the same master keys and activate the P11-MK or RCS-MK. For security reasons, you may then want to change all the master keys.

Note: A TKE workstation is required to load key parts for the P11 master key. See "Entering master key parts using the TKE workstation" on page 159 and the *z/OS Cryptographic Services ICSF TKE Workstation User's Guide* for details.

Setting the Master Key

After the master keys have been cleared, reenter the same master keys by following these steps:

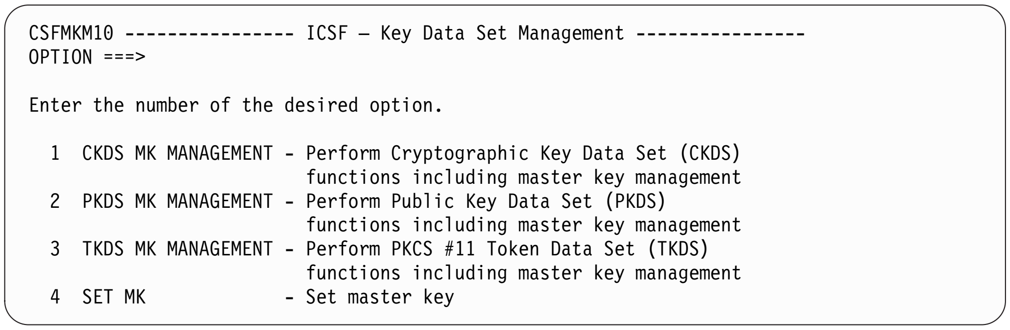
1. Load the new P11 master key using the TKE workstation or your new RCS master key using your RCS vendor utility.
2. Commit the new P11 master key using the TKE workstation or your new RCS master key using your RCS vendor utility.
3. To activate the P11, RCS, or both P11 and RCS master keys you just entered, you need to set it. On the ICSF Primary Menu panel, select option 2, KDS MANAGEMENT.



```
HCR77C0 ------------ Integrated Cryptographic Service Facility ---------

OPTION ===>

Enter the number of the desired option.

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

Figure 44. Selecting KDS MANAGEMENT on the ICSF primary menu panel

4. The Key Data Set Management panel appears. To set the P11, RCS, or both P11 and RCS master keys, select option 4, SET MK.



```
CSFMKM10 --------------- ICSF – Key Data Set Management ---------------
OPTION ===>

Enter the number of the desired option.

  1  CKDS MK MANAGEMENT - Perform Cryptographic Key Data Set (CKDS)
                          functions including master key management
  2  PKDS MK MANAGEMENT - Perform Public Key Data Set (PKDS)
                          functions including master key management
  3  TKDS MK MANAGEMENT - Perform PKCS #11 Token Data Set (TKDS)
                          functions including master key management
  4  SET MK             - Set master key
```

Figure 45. Selecting SET MK on the Key Data Set Management panel

After you select option 4, ICSF checks that the states of the registers are correct. ICSF then transfers the P11, RCS, or both P11 and RCS master keys from the new master key register to the current master key register. This process sets the master key. When ICSF attempts to set the master key, it displays a message on the top right of the Key Data Set Management panel. The message indicates either that the master key was successfully set or that an error prevented the completion of the set process.

You can now change the P11, RCS, or both master key or keys, if you choose to, for security reasons. Continue with "Changing the Master Key" on page 162.

Chapter 9. Key management on systems without coprocessors

The CKDS can be used to manage clear AES and DES DATA keys on a system that does not have any cryptographic coprocessors or accelerators.

Initializing the CKDS at first-time startup

There are three formats of the CKDS. All formats are supported for systems without coprocessors.

- A fixed length record format supported by all releases of ICSF.
- A variable length record format supported by ICSF FMID HCR7780 and later releases.
- A common variable length record format (KDSR) supported by ICSF FMID HCR77A1 and later releases.

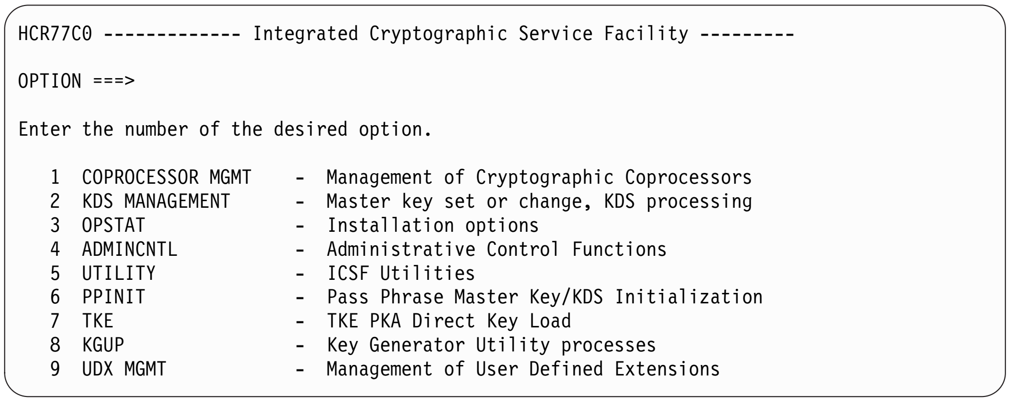
A CKDS is not required in order to use ICSF. However, by defining and initializing a CKDS, secure CCA symmetric key functions are available, and ICSF can be used to manage CCA symmetric key tokens in the CKDS. When you initialize a CKDS, you can copy the disk copy of the CKDS to create other CKDSs for use on the system. You can also use a CKDS from another ICSF system.

At any time, you can read a different disk copy into storage. For information about how to read a disk copy into storage, see "CKDS refresh" on page 169.

A CKDS initialized on a system without coprocessors cannot be used with a system with coprocessors. ICSF will terminate during initialization and issue the CSFM128E message if you attempt to start ICSF with a CKDS that was initialized on a system without coprocessors. The CKDS cannot be updated to support systems with coprocessors.

Steps for initializing a CKDS

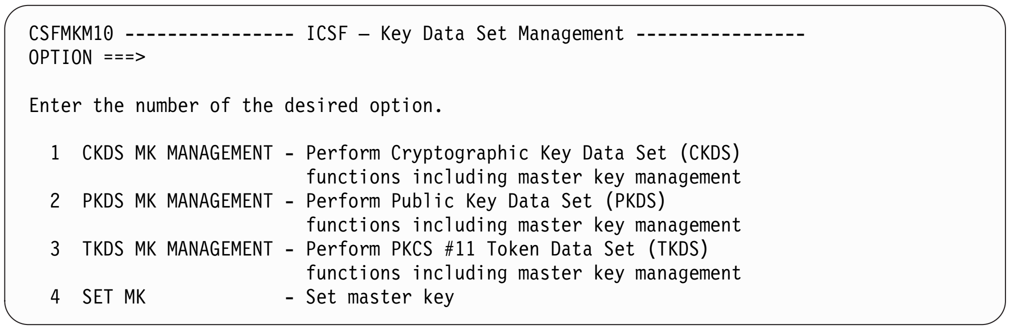
1. From the ICSF Primary Menu, select option 2, KDS MANAGEMENT.



```
HCR77C0 ------------- Integrated Cryptographic Service Facility ---------

OPTION ===>

Enter the number of the desired option.

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

Figure 46. Selecting KDS MANAGEMENT on the ICSF primary menu panel

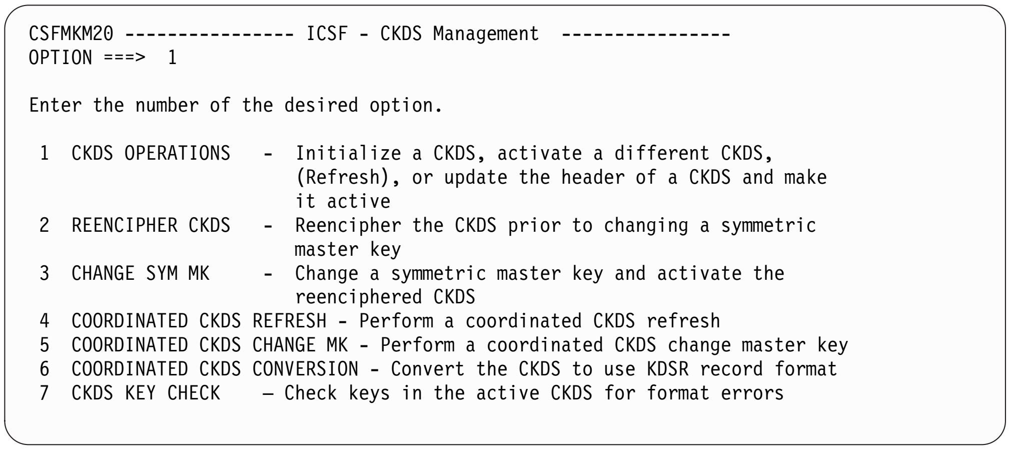
2. The Key Data Set Management panel appears. Select option 1, CKDS MK MANAGEMENT.



```
CSFMKM10 ---------------- ICSF – Key Data Set Management ----------------
OPTION ===>

Enter the number of the desired option.

  1  CKDS MK MANAGEMENT - Perform Cryptographic Key Data Set (CKDS)
                          functions including master key management
  2  PKDS MK MANAGEMENT - Perform Public Key Data Set (PKDS)
                          functions including master key management
  3  TKDS MK MANAGEMENT - Perform PKCS #11 Token Data Set (TKDS)
                          functions including master key management
  4  SET MK             - Set master key
```

Figure 47. Selecting CKDS MK MANAGEMENT on the Key Data Set Management panel

3. The CKDS Management panel appears. Select option 1, CKDS OPERATIONS.

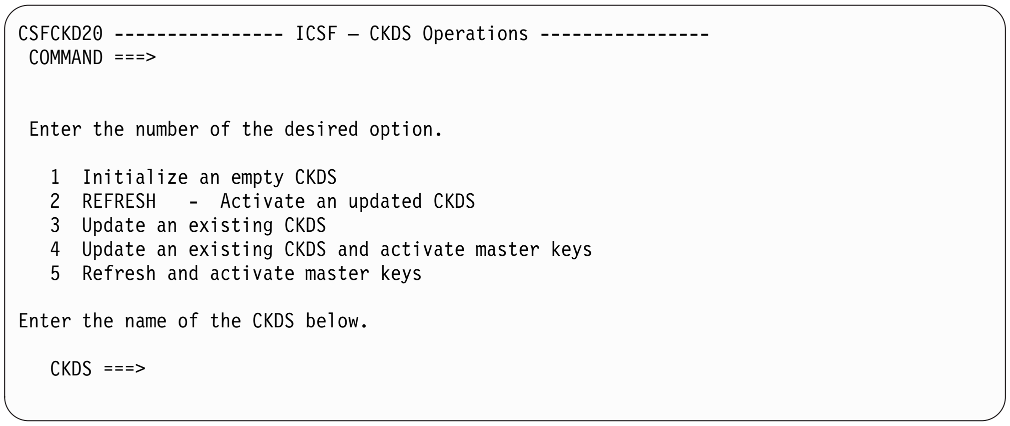


```
CSFMKM20 ---------------- ICSF - CKDS Management  ----------------
OPTION ===>  1

Enter the number of the desired option.

 1  CKDS OPERATIONS   -  Initialize a CKDS, activate a different CKDS,
                         (Refresh), or update the header of a CKDS and make
                         it active
 2  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing a symmetric
                         master key
 3  CHANGE SYM MK     -  Change a symmetric master key and activate the
                         reenciphered CKDS
 4  COORDINATED CKDS REFRESH - Perform a coordinated CKDS refresh
 5  COORDINATED CKDS CHANGE MK - Perform a coordinated CKDS change master key
 6  COORDINATED CKDS CONVERSION - Convert the CKDS to use KDSR record format
 7  CKDS KEY CHECK    – Check keys in the active CKDS for format errors
```

Figure 48. Selecting CKDS OPERATIONS on the CKDS Management panel

4. The CKDS Operations panel appears. In the CKDS field, enter the name of the empty VSAM data set that was created to use as the disk copy of the CKDS.
   - The name you enter can be the same name that is specified in the CKDSN keyword option in the installation options data set. You can also initialize a data set that might serve as a backup. For information about creating a CKDS and specifying the CKDS name in the installation options data set, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

   Select option 1, Initialize an empty CKDS.



```
CSFCKD20 ---------------- ICSF – CKDS Operations ----------------
 COMMAND ===>


 Enter the number of the desired option.

   1  Initialize an empty CKDS
   2  REFRESH   -  Activate an updated CKDS
   3  Update an existing CKDS
   4  Update an existing CKDS and activate master keys
   5  Refresh and activate master keys

Enter the name of the CKDS below.

   CKDS ===>
```

Figure 49. The CKDS Operations panel

ICSF creates the header record in the disk copy of the CKDS and refreshes the CKDS.

When ICSF completes all these steps, the message INITIALIZATION COMPLETE appears.

CKDS refresh

When you initialize a CKDS for the first time, you can copy the disk copy of the CKDS to create other CKDSs for the system. You can use the dynamic CKDS update callable services to add or update the disk copy of the current in-storage CKDS. For information on using the dynamic CKDS callable services, refer to the z/OS Cryptographic Services ICSF Application Programmer's Guide.

Refreshing to the same CKDS or a different CKDS in the sysplex can be done using the Coordinated CKDS Refresh utility. All members of the sysplex sharing the CKDS will load the specified CKDS and make it the active CKDS. See “Performing a coordinated CKDS refresh” on page 136 for additional information.

The CKDS can be refreshed on a single system using the local CKDS Refresh utility. The utility can invoked from the ICSF panels or using the CSFEUTIL. See “Performing a local CKDS refresh” on page 135.

A coordinated CKDS refresh can be done by writing an application to invoke the Coordinated KDS Administration (CSFCRC) callable service. See “Performing a coordinated CKDS refresh” on page 136 for more information.

Callable services

These callable services can be used on a system without coprocessors with an initialized CKDS. The key management services can only be used to manage clear keys, encrypted keys cannot be managed in this configuration.

- Key Data Set List (CSFKDSL)
- Key Data Set Metadata Read (CSFKDMR)
- Key Data Set Metadata Write (CSFKDMW)
- Key Record Create (CSNBKRC) and Key Record Create2 (CSNBKRC2)
- Key Record Delete (CSNBKRD)
- Key Record Read (CSNBKRR) and Key Record Read2 (CSNBKRR2)

  Key Record Read will not return a clear key token to the caller unless the caller is in supervisor state or system key.

- Key Record Write (CSNBKRW) and Key Record Write2 (CSNBKRW2)

These services support labels for the key identifier:
- Field Level Decipher (CSNBFLD)
- Field Level Encipher (CSNBFLE)
- Symmetric Key Decipher (CSNBSYD)
- Symmetric Key Encipher (CSNBSYE)
- Symmetric MAC Generate (CSNBSMG)
- Symmetric MAC Verify (CSNBSMV)

These services do not require a coprocessor:
- Key Token Build (CSNBKTB)
- Key Token Build2 (CSNBKTB2)
- MDC Generate (CSNBMDG)
- One Way Hash (CSNBOWH)
- Random Number Generate (CSNBRNG)
- Random Number Generate Long (CSNBRNGL)

Chapter 10. Running in a Sysplex Environment

ICSF is supported in a SYSPLEX environment. The CKDS, PKDS, and TKDS can be shared across systems in a sysplex. This topic discusses sharing and managing key data sets and managing master keys in a sysplex.

Sysplex communication level

There are several communication protocols for sysplex support for the key data set. ICSF uses the level protocol of the lowest release of ICSF of the systems that are sharing a particular key data set. The newer protocols provide performance enhancements for update processing in a sysplex environment.

For the coordinated administration utilities, the minimum level is 2. This is the level that was introduced in ICSF FMID HCR7790 for the CKDS and in ICSF FMID HCR77A0 for the PKDS and TKDS. All systems in the sysplex, regardless of their active key data set, must be at the minimum level or higher in order for ICSF to use this new protocol.

To determine what protocol ICSF is using, check the ICSF job log for the following message:

```
CSFM639I ICSF COMMUNICATION LEVEL FOR CKDS CHANGED FROM 0 TO 3.
```

If you are using ICSF FMID HCR77B1 or later, the DISPLAY ICSF operator command can be used to display current communication level.

```
D ICSF,KDS
CSFM668I 11.18.28 ICSF KDS 534
CKDS ISFTEST.SHERID.CKDSNEW
FORMAT=KDSR COMM LVL=3 SYSPLEX=Y MKVPs=DES AES
PKDS ISFTEST.SHERID.PKDSNEW
FORMAT=KDSR COMM LVL=3 SYSPLEX=Y MKVPs=RSA ECC
TKDS ISFTEST.SHERID.TKDSNEW
FORMAT=KDSR COMM LVL=3 SYSPLEX=Y MKVPs=P11 RCS
```

Note: The ICSF communication protocol is not configurable. ICSF determines the communication protocol internally based on the ICSF instances that have joined the sysplex group. See the CSFM639I message description in *z/OS Cryptographic Services ICSF Messages* for details.

Coordinated change master key and coordinated refresh utilities

There are utilities that coordinate key data set refreshes and master key changes across sysplex members sharing the same active key data set. The coordinated administration functions simplify key data set management by automating the manual process for performing local refreshes and local master key changes. Although a sysplex environment is not required to use these functions, sysplex environments gain the maximum benefit from them when the changes are coordinated across all LPARs sharing the same active key data set.

The utilities are initiated from a single ICSF LPAR. This LPAR drives the operation across the sysplex by using sysplex messaging to other members sharing the same active key data set. Only one coordinated administration function may be performed at a time.

Coordinated KDS refresh

The coordinated KDS refresh utility, for the CKDS and PKDS only, drives the initiating system to send sysplex messages to all sysplex members sharing the same active key data set, instructing them to either refresh their in-store key data set copy of the active key data set, or refresh their in-store key data set copy to a new key data set. Performing a coordinated refresh to a new key data set results in the new key data set becoming the active key data set for all sysplex members in this key data set sysplex cluster.

Coordinated KDS change master key

Important: The master keys are loaded into the new master key register. The coordinated KDS change master key utility that is described in this section requires the master key to be in the new master key register and not set. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. This utility fails if the master key is not in the new master key register.

The coordinated KDS change master key utility, for the CKDS, PKDS, and TKDS, reenciphers the active key data set disk-copy to a new key data set using the master key values into the new master key registers. Before performing the coordinated change master key function, you must load the new master key registers.

For CKDS, the coordinated change master key utility can be used to change the DES master key, the AES master key, or both.

For PKDS, the coordinated change master key utility can be used to change the RSA master key, the ECC master key, or both.

For TKDS, the coordinated change master key utility is the only function available for changing the P11 and RCS master keys. It can be used to change the P11 master key, the RCS master key, or both.

After reenciphering the active key data set disk-copy, the initiating system will send sysplex messages to the other members sharing the same active key data set to inform them to re-load their in-store key data set from the new reenciphered key data set. Next, the initiating system sets the master keys for the new master key registers and make the new key data set the active key data set.

Finally, the initiating system sends sysplex messages to the other members of the key data set sysplex cluster to inform them to set their master keys for the new master key registers and to make the new key data set their active key data set.

Dynamic KDS update controls

When performing a coordinated change master key on the CKDS, PKDS, or TKDS, it is not required to disable dynamic KDS updates within the sysplex while performing a coordinated change master key. This is an enhancement over the local master key change functions, for which disallowing dynamic KDS update services is recommended.

During a coordinated change master key, dynamic key data set update requests are routed to, and processed by, the ICSF instance that initiated the coordinated change master key. The initiator processes dynamic key data set updates against the active key data set during the coordinated change master key. When the initiating system has reenciphered the key data set, and before it coordinates the key data set master key change across the sysplex, there is a brief suspension to dynamic KDS update processing. During this brief suspension, dynamic key data set updates that were processed by the initiator are applied to the new reenciphered key data set.

If you cannot tolerate a temporary suspension of dynamic KDS update services in your workload while processing a coordinated change master key and would prefer that update requests are failed instead, you should disallow dynamic KDS access prior to performing coordinated change master key.

During a coordinated TKDS change master key, all PKCS #11 session objects are reenciphered on all members of the TKDS sysplex cluster. The PKCS #11 session objects are reenciphered after the TKDS records are reenciphered and right before the new P11 or RCS master key value is set during the temporary suspensions of dynamic TKDS update services.

For a coordinated CKDS and PKDS refresh, dynamic KDS update processing is internally suspended by the initiator until the coordinated refresh completes. However, IBM still recommends that you disallow dynamic access prior to performing a coordinated refresh.

Key store policy

If a Key Store Policy is defined on the active CKDS, the active PKDS, or both the active CKDS and PKDS, it will continue to be used on the new CKDS and the new PKDS after a coordinated change master key or coordinated refresh completes. The TKDS does not have a Key Store Policy.

Initializing ICSF for the first time in a sysplex

CCA

The CCA master keys must be loaded and the CKDS and PKDS must be initialized before your CCA applications can run on your sysplex. Detailed procedures are given in the sections on the CKDS and PKDS management. These are the general procedures used to set up ICSF for the first time in a sysplex.

Using the TKE workstation or master key entry panels

You can select the master keys you want to use with your applications when using the TKE workstation or the Master Key Entry panels.

1. Start ICSF on all LPARs in your sysplex. All LPARs should be using the same installation options data set.
2. Load the new master key registers for the master keys you are going to use on all LPARs. This can be done by using the ICSF Master Key Entry panels or the TKE workstation.
3. Initialize your CKDS and PKDS on one LPAR.
4. On all other LPARs, you can stop and start ICSF. This will load the initialized key data sets and set the master keys and your coprocessors will be active and available for work.

Note: While the key data sets have been initialized on the first LPAR, the in-store copy of the key data sets on the other LPARs in the sysplex have not be refreshed and the header record is not in the in-store copies. Stopping and starting ICSF reloads the key data sets and sets the master keys.

Using pass phrase initialization utility

You can use the Pass phrase Initialization utility (PPINIT) to initialize the CKDS and PKDS and load all the CCA master keys available on your system. You use PPINIT to initialize the CKDS and PKDS on one LPAR and use the pass phrase and initialized CKDS and PKDS to load the master keys on the remaining LPARs.

1. Start ICSF in the first LPAR and follow the instructions in Chapter 6, "Using the pass phrase initialization utility," on page 95.
2. Once the first LPAR has been successfully initialized, start ICSF in the other LPARs that are sharing the same CKDS and PKDS.
3. From each LPAR that is sharing the same CKDS and PKDS, go to the Pass Phrase Initialization panel. See "Steps for reinitializing a system" on page 99.
   a. Enter the same pass phrase as entered on the first LPAR.
   b. Select 'Reinitialize System'.
   c. Enter the same CKDS name and PKDS name as entered on the first LPAR.

PKCS #11

The TKDS must be initialized before your PKCS #11 applications can be run on your sysplex and the P11 or RCS master keys must be loaded for PKCS #11 secure key applications. Detailed procedures are given in the sections on the TKDS management. These are the general procedures used to set up ICSF for the first time in a sysplex.

1. Start ICSF on all LPARs in your sysplex. All LPARs should be using the same installation options data set.
2. Load the new P11 master key registers for the master keys you are going to use on all LPARs using the TKE workstation or load the new RCS master key using your RCS vendor utility.
3. Initialize your TKDS on one LPAR.
4. On all other LPARs, you can stop and start ICSF. This will set the master keys and your coprocessors will be active and available for work.

Note: While the key data sets have been initialized on the first LPAR, the in-store copy of the key data sets on the other LPARs in the sysplex have not be refreshed and the header record is not in the in-store copies. Stopping and starting ICSF reloads the key data sets and sets the master keys.

CKDS management in a sysplex

ICSF may share the same active CKDS across multiple LPARs on the same zSeries server or across LPARs on different zSeries servers. All ICSF instances sharing the same active CKDS must have the same symmetric master keys installed.

It is not required that all ICSF instances share their active CKDS across a sysplex. It is also not required that all ICSF instances in a sysplex be configured with the same active CKDS. Each system may have its own master keys and its own active CKDS. A sysplex may have a combination of ICSF instances that share their active CKDS and ICSF instances that do not share their active CKDS.

In a sysplex environment, a set of ICSF instances all sharing the same active CKDS can be described as a CKDS sysplex cluster. Other ICSF instances configured with different active CKDSs can join the same sysplex group to create multiple CKDS sysplex clusters.

It is not required for each of the ICSF instances sharing the same active CKDS to be configured with the same DOMAIN. Cryptographic Coprocessor DOMAINs may be split up across LPARs all sharing the same active CKDS.

The CKDS can be refreshed using these utilities:

- Local CKDS Refresh – Refreshes the in-storage copy of the specified CKDS on the LPAR where it is initiated. See "Performing a local CKDS refresh" on page 135 or Chapter 17, "Using the ICSF Utility Program CSFEUTIL," on page 343.
- Coordinated CKDS refresh – Refreshes the in-storage copy of the specified CKDS on all members of the sysplex sharing the CKDS. See "Performing a coordinated CKDS refresh" on page 136.

When sharing the CKDS, a few precautions should be observed:

- Dynamic CKDS services update the DASD copy of the active CKDS and the in-storage copy on the system where it is run. The SYSPLEXCKDS option in the ICSF installation options data set provides consistent sysplex-wide update of the DASD copy of the active CKDS and the in-storage copies of the active CKDS for all members of the sysplex sharing the same active CKDS. If SYSPLEXCKDS(YES,FAIL(*xxx*)) is specified in the installation options data set, sysplex messages will be issued to sysplex members configured with the same active CKDS. The messages will inform them of the CKDS update and request them to update their in-storage CKDS copy. If SYSPLEXCKDS(NO,FAIL(*xxx*)) is specified in the installation options data set, sysplex messages will not be sent to sysplex members for CKDS updates. When configured this way, either a coordinated refresh or a local CKDS refresh must be performed to load the updates into ICSF's in-storage copy of the CKDS.
- If multiple sysplexes share a CKDS, or if a sysplex and other non-sysplex systems share a CKDS, there is no provision for an automatic update of the in-storage copies of the CKDS on the systems that are not in the same sysplex as the system initiating the CKDS update. When configured this way, either a coordinated CKDS refresh or a local CKDS refresh will be required on the systems that are sharing the same active CKDS, but are not in the same sysplex as the initiating system in order to update the in-storage copy on each system.
- If KGUP is used to update the active CKDS, the update is only made to the DASD copy of the CKDS. Either a coordinated CKDS refresh or a local CKDS refresh must be performed to load the updates into ICSF's in-storage copy of the CKDS.
- There are three formats of the CKDS. Sysplex members must be running an ICSF FMID that supports the format being shared.
  - Fixed-length record format supported by all releases of ICSF.
  - Variable-length record format supported by ICSF FMID HCR7780 and later releases.
  - Common variable-length format record (KDSR) supported by ICSF FMID HCR77A1 and later releases.

Setting symmetric master keys for the first time when sharing a CKDS in a sysplex environment

Setting symmetric master keys for the first time in a sysplex environment can be accomplished using:

- The optional TKE workstation (Group of coprocessors and/or group of domains function). See *z/OS Cryptographic Services ICSF TKE Workstation User's Guide* for more information.

- Master Key Entry panels.
- The pass phrase initialization utility (PPINIT).

Before setting symmetric master keys for the first time in a sysplex environment, you will need to allocate an empty CKDS. For information about defining a CKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Once you have allocated an empty CKDS, all LPARs that will share this CKDS must update their ICSF options data set to use this CKDS as their active CKDS. On the first LPAR that starts ICSF, you will load the symmetric master keys, initialize the CKDS, and set the symmetric master keys. On all other LPARs that will share the same active CKDS, you will only load the same master keys, refresh the CKDS, and then set the master key. You will only initialize the CKDS once from the first LPAR.

Using the TKE workstation or master key entry panels

Master key entry may be used to set master keys in a sysplex environment.

1. Load your master keys in the first LPAR as described in "Entering master key parts" on page 111 or use the TKE workstation.
2. Initialize the CKDS from the first LPAR as described in "Steps for initializing a CKDS" on page 129.
3. For all remaining LPARs, load the master keys and do one of the following:
   - Stop and start ICSF.
   - Refresh the CKDS using the local CKDS refresh utility and set the master keys using the SET MK utility.
   - Use the local Symmetric change master key utility.
   - Use the Refresh and activate master key option on CKDS Operations panel on each LPAR.

Using pass phrase initialization utility

The Pass Phrase Initialization utility can be used to set the CCA master keys and initialize the CKDS and PKDS in a sysplex environment.

1. Start ICSF in the first LPAR and follow the instructions in Chapter 6, "Using the pass phrase initialization utility," on page 95.
2. Once the first LPAR has been successfully initialized, start ICSF in the other LPARs that are sharing the same active CKDS.
3. From each LPAR that is sharing the same active CKDS, go to the Pass Phrase Initialization panel and:
   a. Enter the same pass phrase as entered on the first LPAR.
   b. Select 'Reinitialize System'.
   c. Enter the same CKDS name and PKDS name as entered on the first LPAR.

These steps will load and set the same master keys as in the first LPAR and activate the same CKDS.

Updating the CKDS with additional master keys in a sysplex environment

When you add a new master key to your system for your applications, you need to update the key data set so keys can be added to the key data set and ICSF knows the new master key is required for coprocessor activation.

The master keys are loaded into the new master key register. The utilities described in this section require that the master key be in the new master key register. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. The Update Key Data Set utilities will fail if the master key is not in the new master key register.

The procedure to update a key data set in a sysplex:

1. Load the new master key register for the missing master key. The missing master key must be loaded on all active coprocessors on all members of the sysplex sharing the key data set.
2. On one system, update the header record of the key data sets for the master keys you are adding. See "CKDS" on page 132.
3. After the CKDS has been updated and the master key set on the first system, set the new master key on the remaining systems by using option 5, 'Refresh and activate master keys', on the CKDS Operations panel. See "CKDS" on page 132.

Refreshing the CKDS in a sysplex environment

Refreshing to the same CKDS or a different CKDS in the sysplex can be done using the Coordinated CKDS Refresh utility. All members of the sysplex sharing the CKDS will load the specified CKDS and make it the active CKDS. See "Performing a coordinated CKDS refresh" on page 136 for additional information.

The CKDS can be refreshed in a sysplex using the local CKDS Refresh utility. The utility must be run individually on each LPAR in the sysplex, either from the ICSF panels or using the CSFEUTIL utility. See "Performing a local CKDS refresh" on page 135 for additional information.

Changing symmetric master keys in a sysplex environment

Changing the master keys consists of:

1. Loading the new master key registers on all coprocessors on all member of the sysplex sharing the active CKDS.
2. Allocating a new CKDS.
3. Reenciphering the CKDS.
4. Setting the master keys and making the reenciphered CKDS the active CKDS.

Changing the master keys in a sysplex can be done by using the coordinated CKDS change master key utility. After loading the new master keys and allocating the new CKDS, the utility is initiated on one LPAR and all members of the sysplex sharing the CKDS will participate. The CKDS will be reenciphered on the initiating LPAR, all members will refresh to the reenciphered CKDS, and set the master keys. See "Symmetric master keys and the CKDS" on page 142 for additional information.

Changing the master keys in a sysplex using the local change symmetric master key utility is more complicated. See "Steps for reenciphering the CKDS and performing a local symmetric master key change" on page 143 for additional information.

1. Load the new master keys.
2. Allocate a new CKDS.
3. Reencipher the CKDS on one LPAR using the REENCIPHER CKDS utility.

4. Change the master keys on all LPARs individually using the CHANGE SYM MK utility.

PKDS management in a sysplex

The systems sharing a PKDS may be different LPARs on the same zSeries server or different systems across multiple zSeries servers. The only requirement for sharing the PKDS is that the same asymmetric master keys be installed on all systems sharing that PKDS. It is not required to share the PKDS across a sysplex. Each system may have its own asymmetric master keys and its own PKDS. A sysplex may have a combination of systems that share a PKDS and individual systems with separate PKDSs.

In a sysplex environment, a set of ICSF instances all sharing the same active PKDS can be described as a PKDS sysplex cluster. Other ICSF instances configured with different active PKDSs can join the same sysplex group to create multiple PKDS sysplex clusters.

It is not required for each ICSF instances sharing the same active PKDS to be configured with the same DOMAIN. Cryptographic Coprocessor DOMAINs may be split up across LPARs all sharing the same active PKDS.

The PKDS can be refreshed using these utilities:

- Local PKDS Refresh – Refreshes the in-storage copy of the specified PKDS on the system where it is initiated. See "Performing a local PKDS refresh" on page 138 or Chapter 18, "Using the ICSF Utility Program CSFPUTIL," on page 353.
- Coordinated PKDS refresh – Refreshes the in-storage copy of the specified PKDS on all members of the sysplex sharing the PKDS. See "Performing a coordinated PKDS refresh" on page 138.

When sharing the PKDS, a few precautions should be observed:

- Dynamic PKDS services update the DASD copy of the active PKDS and the in-storage copy on the system where it is run. The SYSPLEXPKDS option in the ICSF installation options data set provides consistent sysplex-wide update of the DASD copy of the active PKDS and the in-storage copies of the active PKDS for all members of the sysplex sharing the same active PKDS. If SYSPLEXPKDS(YES,FAIL(*xxx*)) is specified in the installation options data set, sysplex messages will be issued to sysplex members configured with the same active PKDS. The messages will inform them of the PKDS update and request them to update their in-storage PKDS copy. If SYSPLEXPKDS(NO,FAIL(*xxx*)) is specified in the installation options data set, sysplex messages will not be sent to sysplex members for PKDS updates. When configured this way, either a coordinated refresh or a local refresh must be performed to load the updates into ICSF's in-storage copy of the PKDS.
- If multiple sysplexes share a PKDS, or if a sysplex and other non-sysplex systems share a PKDS, there is no provision for an automatic update of the in-storage copies of the PKDS on the systems that are not in the same sysplex as the system initiating the PKDS update. When configured this way, either a coordinated PKDS refresh or a local PKDS refresh will be required on the systems that are sharing the same active PKDS, but are not in the same sysplex as the initiating system in order to update the in-storage copy on each system.
- The PKDS must be initialized for PKA callable services to be enabled. Use the TSO panels to initialize a new PKDS.

- There are two formats of the PKDS. Sysplex members must be running an ICSF FMID that supports the format being shared:
  – Variable-length record supported by all releases of ICSF.
  – Common variable-length format record (KDSR) supported by ICSF FMID HCR77A1 and later releases.

Setting asymmetric master keys for the first time when sharing a PKDS in a sysplex environment

Setting asymmetric master keys for the first time in a sysplex environment can be accomplished using:

- The optional TKE workstation (Group of coprocessors and/or group of domains function). See *z/OS Cryptographic Services ICSF TKE Workstation User's Guide* for more information.
- Master Key Entry panels.
- The pass phrase initialization utility (PPINIT).

Before setting asymmetric master keys for the first time in a sysplex environment, you will need to allocate an empty PKDS. For information about defining a PKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Once you have allocated an empty PKDS, all LPARs that will share this PKDS must update their ICSF options data set to use this PKDS as their active PKDS. On the first LPAR that starts ICSF, you will load the asymmetric master keys, initialize the PKDS, and set the asymmetric master keys. On all other LPARs that will share the same active PKDS, you will only load the same master keys and then set the master key. You should only initialize the PKDS once from the first LPAR that started ICSF.

Using the TKE workstation or master key entry panels

Master key entry may be used to set master keys in a sysplex environment.

1. Load your master keys in the first LPAR as described in "Entering master key parts" on page 111 or use the TKE workstation.
2. Initialize the PKDS from the first LPAR as described in "Steps for initializing the PKDS" on page 131.
3. For all remaining LPARs, load the master keys and do one of the following:
   - Stop and start ICSF.
   - Refresh the PKDS using the local PKDS refresh utility and set the master keys using the SET MK utility.
   - Use the local symmetric change master key utility.
   - Use the refresh and activate master key option on PKDS Operations panel on each LPAR.

Using pass phrase initialization utility

The pass phrase initialization utility can be used to set the CCA master keys and initialize the CKDS and PKDS in a sysplex environment.

1. Start ICSF in the first LPAR and follow the instructions in Chapter 6, "Using the pass phrase initialization utility," on page 95.
2. Once the first LPAR has been successfully initialized, start ICSF in the other LPARs that are sharing the same active PKDS.

3. From each LPAR that is sharing the same active PKDS, go to the Pass Phrase Initialization panel and:
   a. Enter the same pass phrase as entered on the first LPAR.
   b. Select 'Reinitialize System'.
   c. Enter the same CKDS name and PKDS name as entered on the first LPAR.

These steps will load and set the same master keys as in the first LPAR and activate the same PKDS.

Updating the PKDS with additional master keys in a sysplex environment

When you add a new master key to your system for your applications, you need to update the key data set so keys can be added to the key data set and ICSF knows the new master key is required for coprocessor activation.

The master keys are loaded into the new master key register. The utilities described in this section require the master key to be in the new master key register. Newer versions of the TKE workstation allow the master keys to be set from the TKE workstation. The Update Key Data Set utilities will fail if the master key is not in the new master key register.

The procedure to update a key data set in a sysplex:
1. Load the new master key register for the missing master key. The missing master key must be loaded on all active coprocessors on all members of the sysplex sharing the key data set.
2. On one system, update the header record of the key data sets for the master keys you are adding. See “PKDS” on page 133.
3. After the PKDS has been updated and the master key set on the first system, set the new master key on the remaining systems by using option 5, 'Refresh and activate master keys', on the PKDS Operations panel. See “PKDS” on page 133.

Refreshing the PKDS in a sysplex environment

Refreshing to the same PKDS or a different PKDS in the sysplex can be done using the Coordinated PKDS Refresh utility. All members of the sysplex sharing the PKDS will load the specified PKDS and make it the active PKDS. See “Performing a coordinated PKDS refresh” on page 138 for additional information.

The PKDS can be refreshed in a sysplex using the local PKDS Refresh utility. The utility must be run individually on each LPAR in the sysplex, either from the ICSF panels or using the CSFPUTIL utility. See “Performing a local PKDS refresh” on page 138 for additional information.

Changing asymmetric master keys in a sysplex environment

Changing the master keys consists of:
1. Loading the new master key registers on all coprocessors on all member of the sysplex sharing the active PKDS.
2. Allocating a new PKDS.
3. Reenciphering the PKDS.
4. Setting the master keys and making the reenciphered PKDS the active PKDS.

Changing the master keys in a sysplex can be done by using the coordinated PKDS change master key utility. After loading the new master keys and allocating the new PKDS, the utility is initiated on one LPAR and all members of the sysplex sharing the PKDS will participate. The PKDS will be reenciphered on the initiating LPAR, all members will refresh to the reenciphered PKDS, and sets the master keys. See "Asymmetric master keys and the PKDS" on page 145 for additional information.

Changing the master keys in a sysplex using the local change symmetric master key utility is more complicated. See "Steps for reenciphering the PKDS and performing a local asymmetric master key change" on page 146 for additional information.

1. Load the new master keys.
2. Allocate a new PKDS.
3. Reencipher the PKDS on one LPAR using the REENCIPHER PKDS utility.
4. Change the master keys on all LPARs individually using the CHANGE ASYM MK utility.

Notes for the RSA master key

If your system is an IBM z9 or IBM z10 server or an IBM z196 or z114 where all CEX3 coprocessors have licensed internal code older than September 2011:

- The PKA callable services control will be active on your system. It will appear on the Administrative Control Functions panel.
- The RSA master key will be set when the final key part is loaded on the Master Key Entry panel. The master key will be in the current master key register. The PKA callable services control must be disabled to load the RSA master key.
- The coordinated PKDS change master key utility can be used if the RSA master key is loaded using the TKE workstation. The RSA master key should not be set using the TKE workstation. The change master key utility will set the master key.

If your system is an IBM z196 or z114 where there is a CEX3 coprocessor with September 2011 or newer licensed internal code or your system is an IBM zEC12, zBC12, or later server:

- The PKA callable services control will not be active on your system. It will not appear on the Administrative Control Functions panel.
- The RSA master key will not be set when the final key part is loaded on the Master Key Entry panel. The master key will be in the new master key register.

TKDS management in a sysplex

The systems sharing a TKDS may be different LPARs on the same zSeries server or different systems across multiple zSeries servers. It is not required to share the TKDS across a sysplex. Each system may have its own TKDS. A sysplex may have a combination of systems that share a TKDS and individual systems with separate TKDSs. There is no requirement that the DOMAINs must be the same to share a TKDS.

In a sysplex environment, a set of ICSF instances all sharing the same active TKDS can be described as a TKDS sysplex cluster. Other ICSF instances configured with different active TKDSs can join the same sysplex group to create multiple TKDS sysplex clusters.

When sharing the TKDS, a few precautions should be observed:

- Dynamic TKDS services update the DASD copy of the active TKDS and the in-storage copy on the system where it is run. The SYSPLEXTKDS option in the ICSF installation options data set provides consistent sysplex-wide update of the DASD copy of the active TKDS and the in-storage copies of the active TKDS for all members of the sysplex sharing the same active TKDS. If SYSPLEXTKDS(YES,FAIL(*xxx*)) is specified in the installation options data set, sysplex messages will be issued to sysplex members configured with the same active TKDS. The messages will inform them of the TKDS update and request them to update their in-storage TKDS copy. If SYSPLEXTKDS(NO,FAIL(*xxx*)) is specified in the installation options data set, sysplex messages will not be sent to sysplex members for TKDS updates.
- If multiple sysplexes share a TKDS, or if a sysplex and other non-sysplex systems share a TKDS, there is no provision for an automatic update of the in-storage copies of the TKDS on the systems which are not in the same sysplex as the system initiating the TKDS update.
- There are two formats of the TKDS. Sysplex members must be running an ICSF FMID that supports the format being shared.
  - Variable-length record supported by all releases of ICSF.
  - Common variable-length format record (KDSR) supported by ICSF FMID HCR77A1 and later releases.

Setting the PKCS #11 master key for the first time when sharing a TKDS in a sysplex environment

Setting the P11 master key for the first time in a sysplex environment can only be accomplished using the TKE workstation (Group of coprocessors and/or group of domains function). See *z/OS Cryptographic Services ICSF TKE Workstation User's Guide* for more information about TKE.

For instructions on how to set the P11 master keys for the first time when sharing a TKDS in a sysplex environement, refer to "First time use of Enterprise PKCS #11 keys" on page 160 in Chapter 8, "Managing PKCS #11 master keys," on page 159.

Setting the RCS master key for the first time in a sysplex environment can only be accomplished using your RCS vendor utility.

Changing PKCS #11 master keys when the TKDS is shared in a sysplex environment

ICSF coordinates TKDS master key changes across sysplex members sharing the same active TKDS. The master key change is initiated from a single ICSF instance. This instance will drive the operation across the sysplex using sysplex messaging to other members sharing the same active TKDS.

A Coordinated TKDS change master key will reencipher the active TKDS disk-copy to a new TKDS using the master key values that have been pre-loaded into the new master key registers. Before performing the coordinated TKDS change master key function, you must use the TKE to load the new P11 master key registers or your RCS vendor utility to load the new RCS master key.

After reenciphering the active TKDS disk-copy, the initiating system will send sysplex messages to the other members sharing the same active TKDS, informing them to re-load their in-store TKDS from the new reenciphered TKDS. Next, the initiating system will set the P11, RCS, or both master key or keys for the new master key registers that have been pre-loaded and make the new TKDS the active TKDS. Finally, the initiating system will send sysplex messages to the other members of their TKDS sysplex cluster, informing them to set their P11, RCS, or both P11 and RCS master keys for the new master key registers that have been pre-loaded and to make the new TKDS their active TKDS.

During a coordinated TKDS master key change, dynamic TKDS update requests will be routed to, and processed by, the ICSF instance that initiated the coordinated TKDS master key change. The initiator will process dynamic TKDS updates against the active TKDS during the coordinated TKDS change master key. When the initiating system has reenciphered the TKDS, and before it coordinates the TKDS master key change across the sysplex, there is a brief suspension to dynamic TKDS update processing. During this brief suspension, dynamic TKDS updates that were processed by the initiator are applied to the new reenciphered TKDS.

Notes:

1. It is not necessary to be in a sysplex to perform a coordinated TKDS change master key. The procedure to change the P11, RCS, or both P11 and RCS master keys on single system images is identical to that of a sysplex environment.
2. In order to perform a coordinated TKDS change master key:
   - If the TKDS does not contain RCS keys, all systems sharing the TKDS within the sysplex must be at ICSF FMID HCR77A0 or later. The system initiating the coordinated TKDS change master key must have at least one active Enterprise PKCS #11 coprocessor.
   - If the TKDS contains a RCS MKVP, all systems sharing the TKDS within the sysplex must be at ICSF FMID HCR77B1 with PTF OA49069 level or later and must have at least one active Enterprise PKCS #11 coprocessor or regional cryptographic server.

See Chapter 8, "Managing PKCS #11 master keys," on page 159 for information on how to perform a coordinated TKDS change master key.

Chapter 11. Managing Cryptographic Keys Using the Key Generator Utility Program

The key generator utility program (KGUP) generates and maintains keys in the cryptographic key data set (CKDS). The CKDS stores DATA keys, MAC keys, PIN keys, transport keys and other AES and DES keys.

Note: There are restrictions on some key types depending on the cryptographic coprocessors available on your systems. These restrictions are listed in the section describing the TYPE keyword.

There are three formats of the CKDS:
- Fixed-length record (supported by all releases of ICSF).
- Variable-length record (supported by ICSF FMID HCR7780 and later releases).
- Common record format (KDSR) (supported by ICSF FMID HCR77A1 and later releases).

All formats are supported by KGUP.

| To run KGUP, ICSF must be active, the user must have access to KGUP (via the
| CSFKGUP profile in the CSFSERV class) and CONTROL authority to the profile
| covering the CKDS in the DATASET class, and the CKDS must be initialized. On systems with cryptographic coprocessors. master keys must be loaded on the cryptographic coprocessors. On systems without coprocessors, for release ICSF FMID HCR77A0 and later, random number can be generated to create clear DES and AES keys.

| In addition, if key lifecycle auditing of tokens or labels is enabled
| [AUDITKEYLIFECKDS(TOKEN(YES)) or AUDITKEYLIFECKDS(LABEL(YES)) is
| specified], the user must have access to the CSFGKF profile in the CSFSERV class
| to generate the key fingerprint used in auditing.

Use the CSFKGUP profile in the CSFSERV class to permit or deny users access to the utility.

You use KGUP to perform these tasks:
- Generate or enter keys
- Maintain CKDS entries by deleting or renaming the entries
- Load completed operational keys into the CKDS that were entered from a TKE workstation.

When KGUP generates or receives a key value, the program either adds a new entry or updates an existing entry in the CKDS. For information about how KGUP generates and receives keys to establish key exchange with other systems, see "Using KGUP for key exchange" on page 187.

Each key that KGUP generates (except clear DES and AES data-encrypting keys) exists in the CKDS enciphered under your system's master key.

You use control statements to specify the functions for KGUP to perform. The control statement specifies the task you want KGUP to perform and information about the CKDS entry that is affected. For example, to have KGUP generate an importer key-encrypting key, you use a control statement like:

```
ADD  LABEL(KEY1)  TYPE(IMPORTER)
```

When KGUP processes the control statement, the program generates a key value and encrypts the value under a master key variant for an importer key-encrypting key. KGUP places the key in a CKDS entry labelled KEY1. The key type field of the entry specifies IMPORTER. For a description of the fields in a CKDS entry, see "Specifying KGUP data sets" on page 224.

You store the control statements in a data set. You must also specify other data sets that KGUP uses when the program processes control statements. You submit a batch job stream to run KGUP. In the job control statements, you specify the names of the data sets that KGUP uses.

KGUP changes a disk copy of the CKDS according to the functions you specify with the control statements. When KGUP changes the disk copy of the CKDS, you may replace the in-storage copy of the CKDS with the disk copy using the ICSF panels. This operation should be performed on all systems sharing the updated CKDS.

To use KGUP, you must perform these tasks:
- Create control statements
- Specify data sets
- Submit a job stream

You may also want to refresh the CKDS with the disk copy of the CKDS that KGUP updated. You can use the KGUP panels to help you perform these tasks. However you can also use KGUP without accessing the panels. This topic first describes each of the tasks to run KGUP, and then describes how to use the panels to perform the tasks.

Steps for disallowing dynamic CKDS updates during CKDS administration updates

ICSF prioritizes changes to the CKDS sequentially, regardless of the source. A KGUP job does not have priority over application calls to the dynamic CKDS update services. Exclusive use of the CKDS by any one application call is minimal, however. For this reason, ICSF allows for a maximum concurrent usage of the CKDS by both KGUP and the dynamic update services.

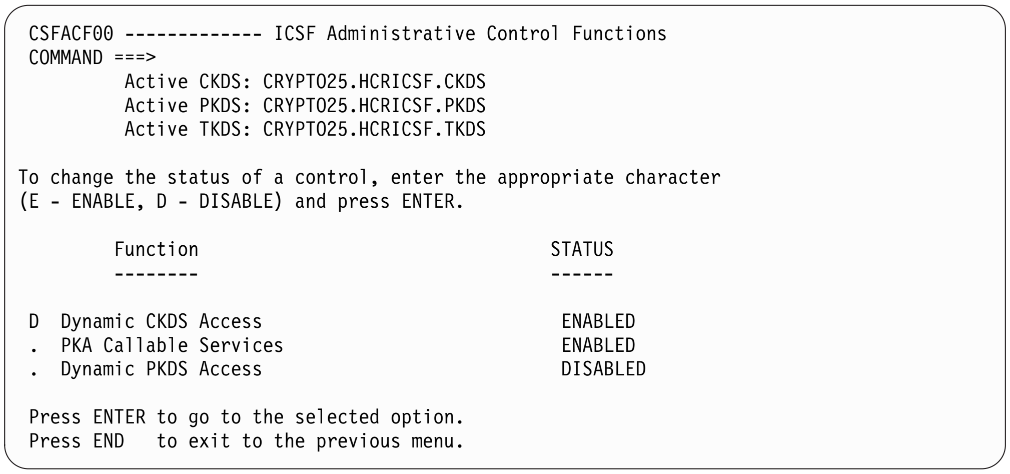
When you perform any function that affects the current CKDS (such as reenciphering, refreshing, or changing the master key), you should consider temporarily disallowing the dynamic CKDS update services.

If you are planning to use KGUP to make significant changes to the CKDS, you should disallow dynamic CKDS update on every system which shares the CKDS. If you are planning to perform a coordinated CKDS change master key or coordinated CKDS refresh operation on a large CKDS (millions of records), you may experience a temporary suspension of CKDS update requests running in parallel. If you cannot tolerate a temporary suspension in your workload, and would prefer that update requests are failed instead of suspended, you should disallow dynamic CKDS updates on every system which shares the same active CKDS prior to performing the coordinated CKDS administration operation. If an application tries to use the dynamic CKDS update services when they are disallowed, the return code indicates that the CKDS management service has been disabled by the system administrator.

To disallow dynamic CKDS access, perform these tasks:

1. Select option 4, Administrative Control Functions, on the “ICSF Primary Menu panel” on page 387.

   The Administrative Control Functions panel appears. See Figure 50.
2. Enter a 'D' to disallow dynamic CKDS access.



```
 CSFACF00 ------------- ICSF Administrative Control Functions
 COMMAND ===>
          Active CKDS: CRYPTO25.HCRICSF.CKDS
          Active PKDS: CRYPTO25.HCRICSF.PKDS
          Active TKDS: CRYPTO25.HCRICSF.TKDS

To change the status of a control, enter the appropriate character
(E - ENABLE, D - DISABLE) and press ENTER.

        Function                                STATUS
        --------                                ------

 D  Dynamic CKDS Access                          ENABLED
 .  PKA Callable Services                        ENABLED
 .  Dynamic PKDS Access                          DISABLED

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

Figure 50. Selecting to Disallow Dynamic CKDS Access on User Control Functions Panel

3. Press ENTER.

   The message CKDS UPDATES DISABLED appears in the upper right-hand corner of the panel.
4. Press END to return to the Primary Menu panel.

Using KGUP for key exchange

KGUP generates keys that are complementary keys. Complementary keys have the same clear key value for corresponding key types. KGUP generates and maintains these types of complementary keys:

- DES Data-encrypting (DATA)
- AES and DES Data-encrypting (CIPHER) and cipher-translate (CIPHERXL) keys
- AES and DES Importer key-encrypting key and exporter key-encrypting key
- DES Input PIN-encrypting key and output PIN-encrypting key
- DES MAC generation key and MAC verification key
- DES PIN generation key and PIN verification key

KGUP generates control information and key tokens for distribution to another site. For DES keys, this information can be used as input to KGUP to import the complementary key into the CKDS. For AES keys using the variable-length key token, KGUP cannot be used to import encrypted key values. Clear key values can be used with AES keys. To import a AES variable-length key token generated by KGUP, the receiving site must be sent the whole key token and must use the

Symmetric Key Import2 callable service to import the key under the AES master key and use the CKDS Key Token Create2 service to store the key in the CKDS.

When you distribute keys or PINs, your system has one key, and the other system has the complementary key. For example, when your system sends a DATA key to another system, the importer and exporter key-encrypting keys at the systems complement each other. The DATA key is encrypted under an exporter key-encrypting key at your system. The DATA key is decrypted by the complementary importer key-encrypting key at the receiving system.

When KGUP generates a key, the other system involved in the key or PIN exchange needs the complement of the key. When KGUP generates a key, the program also generates a control statement to create the complement of the key. You send the control statement to the other system which uses the statement to create the complementary key.

For example, when you use KGUP to create an input PIN-encrypting key, KGUP also creates a control statement for the complementary output PIN-encrypting key. You send the control statement to another system. The other system uses the control statement to create the output PIN-encrypting key. Then your system can send PIN blocks to the other system.

For some key types, you can choose the output key type by specifying the OUTTYPE parameter on a KGUP ADD statement. For example, you can generate a CIPHER key for inclusion into the CKDS and export a copy of the key as either a CIPHER key or a CIPHERXL key. If you export the copy of the CIPHER key as a CIPHER key, the receiver of the key can use it to decipher data. If you export the copy of the CIPHER key as a CIPHERXL key, the receiver can use the key only in the Ciphertext Translate2 callable service to translate cipher text from one cipher translation key to another. The receiver of the CIPHERXL key cannot use the key to actually decipher the data.

KGUP stores the complementary key control statement in a data set. Because some cryptographic systems may not use KGUP control statements, KGUP also stores complementary key information as a record in a different data set. The information is not in the form of a control statement. You process and send the information to a system which creates the complementary key.

When KGUP generates a key, the program also generates information to create the complementary key. This information includes the complementary key value. The value is either a clear key value or encrypted key value. For an encrypted key value, the program encrypts the value under an exporter key. The importer key that complements this exporter key already exists at the other system. The importer key is one key in a complementary transport key pair that your system already established with the other system. The pair would be an importer key on the other system and an exporter key on your system. The other system reenciphers the value from under the importer key to under its master key to generate the complementary key.

Besides generating keys and complementary key information, KGUP imports key values that are sent from other systems. The program can receive a control statement to create a key that is the complement of a key on another system. The key value your KGUP receives may be encrypted under a transport key. The transport key would be one key of a complementary transport key pair that you already established with the other system. The pair would be an exporter key on the other system and an importer key on your system. KGUP reenciphers the complementary key from under the importer key to under the master key and places the key in the CKDS.

CAUTION:
KGUP does not create complementary key control statement for existing key labels, nor new a key label that has CLEAR parm specified in the KGUP statement.

For KGUP to send or receive keys in a key exchange with another system, the systems must previously establish a pair of complementary transport keys. For example, KGUP on one system defines the pair and generates the importer key in the clear. KGUP on the other system uses this value to define a pair of keys that are complements of the keys at the original site. For an example of how two ICSF systems establish pairs of complementary transport keys for key exchange, see "Scenario of Two ICSF Systems Establishing Initial Transport Keys" on page 258.

When a transport key is used, the key value of the key being exported is encrypted by a variant of the transport key. This encrypted key value can be imported into any cryptographic product that supports CCA. However, systems with some cryptographic products do not recognize CCA control vectors. When you exchange keys with such a system, a key that you send or receive is enciphered under a transport key directly. You specify to KGUP a NOCV transport key and the key value will be encrypted by the transport key only.

You can define a pair of complementary transport keys with another system so your system and the other system can exchange keys without control vectors. You use a control statement to indicate to KGUP to produce these keys. Then send the clear value that KGUP produced to the PCF system so the system can generate the corresponding complementary pair of keys. Then you use the transport keys to exchange other keys. Refer to "Scenario of an ICSF System and a PCF System Establishing Initial Transport Keys" on page 260 for an example of how to establish pairs of complementary transport keys for key exchange between an ICSF system and a PCF system.

You can also use KGUP to create complementary keys that are used by two different systems. Neither key would be operational on your system so KGUP would not update your CKDS. When KGUP generates the complementary key information, you send it to the two systems that need to share complementary keys.

Using KGUP control statements

You use control statements to specify the function you want the key generator utility program (KGUP) to perform. You use job control language (JCL) to submit the control statements to KGUP. You can create and submit KGUP control statements either on your own or using the KGUP panels. OPKYLOAD control statements cannot be created using the KGUP panels.

You specify information to KGUP using an ADD, UPDATE, DELETE, RENAME, SET or OPKYLOAD control statement. You use keywords on the control statement to specify:

- The function KGUP performs
- Information about the key that KGUP processes

For example, if you specify the KEY keyword on an ADD control statement, you supply a key which KGUP adds to the CKDS in an entry.

This topic describes the syntax of the control statements with their keywords. Use these rules when interpreting the syntax of the control statements:

- Specify uppercase letters and special characters as shown in the examples.
- Lowercase letters represent keyword values that you must specify.
- A bar (|) indicates a choice (OR).
- Ellipses (...) indicates that multiple entries are possible.
- Braces ({ }) denote choices, one of which you must specify.
- Brackets ([ ]) denote choices, one of which you may specify.

Control statements in the Control Statement Input data set may not be longer than 71 characters including the continuation character.

General Rules for CKDS Records

There are some general rules for creating labels for CKDS key records.

- Each label can consist of up to 64 characters. The first character must be alphabetic or a national character (#, $, @). The remaining characters can be alphanumeric, a national character (#, $, @), or a period (.).
- Labels must be unique for all key types except EXPORTER, IMPORTER, IPINENC, PINGEN, PINVER, and OPINENC
- Labels must be unique for any key record, including transport and PIN keys, created or updated using the dynamic CKDS update services.

KGUP and the dynamic CKDS update services, unless they are modified by user-written exits, check for uniqueness according to these rules prior to making any change to the CKDS.

CKDS record level authentication

ICSF may have an optional record level authentication code that is part of each record in the CKDS. The record level authentication code is used to identify when a record in the CKDS is modified by a program other than ICSF. The record level authentication is enabled when the CKDS is initialized and cannot be changed after the CKDS is initialized. If the CKDS is properly protected using RACF profiles, then unauthorized modification of the CKDS can be prevented.

KGUP detects when ICSF and the CKDS are enabled for record level authentication and performs the necessary processing. When record level authentication is not enabled, KGUP does not perform record level authentication processing.

KGUP Uniqueness Checking

KGUP first checks to see if the label in the control statement matches a label that already exists in the CKDS.

If KGUP is processing an ADD control statement and there is no matching record, KGUP continues processing. Also, if KGUP is processing a RENAME control statement and there is no match for the *new-label* parameter, KGUP continues processing the control statement. If KGUP finds a matching label, KGUP then checks whether the key requires a unique label. If the key does not require a unique label, KGUP continues processing the ADD or RENAME control statement. If the key does require a unique label, KGUP stops processing the control statement and issues a message.

If KGUP is processing an UPDATE or DELETE control statement and there is no matching record, KGUP ends processing and issues an error message. Also, if KGUP is processing a RENAME control statement and there is no match for the *old-label* parameter, KGUP ends processing and issues an error message. If KGUP finds a matching label, KGUP continues processing the UPDATE, DELETE, or RENAME control statement.

Dynamic CKDS Update Services Uniqueness Checking

The dynamic CKDS update services require unique record labels in the CKDS. Each service checks to see if the label in the application call matches a label that already exists in the CKDS. For the Key Record Create service, if there is no matching record in the CKDS, ICSF continues processing the application call. If there is a match, ICSF stops processing and returns a return code and reason code to the application. For the Key Record Write and Key Record Delete services, if there is only one record in the CKDS that matches the label in the application call, ICSF continues processing the application call. If there is more than one matching record in the CKDS, ICSF stops processing and returns a return code and reason code to the application.

Key Store Policy Duplicate Token Checking

When the RACF XFACILIT resource CSF.CKDS.TOKEN.NODUPLICATES is enabled, KGUP will check for duplicate encrypted tokens in the CKDS for ADD and UPDATE control statements. When a duplicate token is found, the processing of that control statement will terminated.

Access Control Points and Key Wrapping

User should note that any enabled access control points that affect key wrapping in ICSF (default wrapping, weak key warning or prohibit weak key wrapping) affect KGUP in the same manner as any application using ICSF callable services.

KGUP and key lifecycle auditing

ICSF may log audit records for actions taken by KGUP. The audit records that are logged are dependent on the AUDITKEYLIFECKDS option setting. If auditing is enabled, ICSF makes additional requests to the Crypto Express coprocessor to generate the key fingerprint. This applies to variable-length tokens and all secure tokens. By default, no audit records are logged. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

Syntax of the ADD and UPDATE control statements

The ADD and UPDATE control statements use the same keywords. The ADD control statement adds new keys to the CKDS. UPDATE changes existing key entries. Use the ADD or UPDATE control statement to specify that KGUP generate a key value or import a key value that you provide.

Refer to Figure 51 on page 192 for the syntax of the ADD and UPDATE control statements.

```
{ADD | UPDATE}

  {LABEL(label1[,...,label64]) | RANGE(start-label,end-label)}

  TYPE(key-type)

  [ALGORITHM(DES|AES)]

  [OUTTYPE(key-type)]

  [TRANSKEY(key-label1[,key-label2]) | CLEAR]

  [NOCV]

  [LENGTH(n)]

  [SINGLE | DOUBLEO]

  [KEY(key-value1[,...,key-value4])]

  [KEYUSAGE(key-usage-value1[,...,key-usage-value2])]

  [DKYGENKYUSAGE(key-usage-value1[,...,key-usage-value2])]
```

Figure 51. ADD and UPDATE Control Statement Syntax

LABEL (label1[,...,label64])

This keyword defines the names of the key entries for KGUP to process within the CKDS. KGUP processes a separate entry for each label. If you specify more than one label on an ADD or UPDATE control statement, the program uses identical key values in each entry.

You must specify at least one key label, and you can specify up to 64 labels with the LABEL keyword. For the general rules about key label conventions and uniqueness, see "General Rules for CKDS Records" on page 190.

On a KGUP control statement, you must specify either the LABEL or RANGE keyword. When you supply a key value on the control statement with the KEY keyword, you must specify the LABEL keyword.

RANGE (start-label, end-label)

This keyword defines the range of the multiple labels that you want KGUP to create or maintain within the CKDS.

The label consists of between 2 and 64 characters that are divided as follows:

- The first 1 to 63 characters are the label base. These characters must be identical on both the start-label and end-label and are repeated for each label in the range. For the general rules about key label conventions and uniqueness, see "General Rules for CKDS Records" on page 190.
- The last 1 to 4 characters form the suffix. The number of digits in the start-label and end-label must be the same, and the characters must all be numeric. These numeric characters establish the range of labels KGUP creates. The start-label numeric value must be less than the end-label numeric value.

KGUP creates a separate CKDS entry for each label including the start and end labels. The program generates a different key value for each entry it creates.

You cannot use the RANGE keyword when you supply a key value to KGUP. Only use RANGE to generate a key value. The RANGE and KEY keywords are mutually exclusive.

On a KGUP control statement, you must specify either the LABEL or RANGE keyword.

TYPE (key-type)
This keyword specifies the type of key you want KGUP to process. You can specify only one key type for each control statement. For EXPORTER, IMPORTER, IPINENC, PINGEN, PINVER, and OPINENC key types, KGUP allows keys with the same labels but different key types. You can specify any of the key types in the following table.

Table 39. Key types

Key Type	Algorithm	Usage	Notes
CIPHER	AES	Data-encrypting key for the CSNBSAD and CSNBSAE services	128-, 192, or 256-bit key
CIPHER	DES	Data-encrypting key for the CSNBDEC and CSNBENC services	Single- or double-length key
CIPHERXI	DES	Input cipher translate key for CSNCTT2 and CSNCTT3 services	Double-length key May not have replicated key values. The September, 2012 or later Licensed Internal Code is required.
CIPHERXL	DES	Input cipher translate key for CSNCTT2 and CSNCTT3 services	Double-length key May not have replicated key values. The September, 2012 or later Licensed Internal Code is required.
CIPHERXO	DES	Output cipher translate key for CSNCTT2 and CSNCTT3 services	Double-length key May not have replicated key values. The September, 2012 or later Licensed Internal Code is required.
CLRAES		Clear AES data-encrypting key for the CSNBSYD and CSNBSYE services	128-, 192, or 256-bit key
CLRDES		Clear DES data-encrypting key for the CSNBSYD and CSNBSYE services	Single-, double-, or triple-length key
DATA	AES, DES	Data-encrypting key for the CSNBDEC, CSNBENC, CSNBSAD, CSNBSAE, CSNBSYD, and CSNBSYE services	Single-, double-, or triple-length key for DES 128-, 192-, or 256-bit key for AES
DATAM	DES	Double-length MAC generation key	Double-length key DOUBLEO not allowed
DATAMV	DES	Double-length MAC verification key	Double-length key DOUBLEO not allowed
DECIPHER	DES	Data-decrypting key for the CSNBDEC service	Single- or double-length key.

Table 39. Key types (continued)

Key Type	Algorithm	Usage	Notes
DKYGENKY*	AES, DES	Diversified key generating key for CSNBDKG and CSNBDKG2 services	Double-length key for DES 128-, 192-, or 256-bit key for AES
ENCIPHER	DES	Data-encrypting key for the CSNBENC service	Single- or double-length key
EXPORTER	AES, DES	Exporter key-encrypting key	Double-length key for DES 128-, 192, or 256-bit key for AES
IMPORTER	AES, DES	Importer key-encrypting key	Double-length key for DES 128-, 192, or 256-bit key for AES
IMPPKA	DES	Limited authority importer key-encrypting key	Double-length key
IPINENC	DES	Input PIN encryption key	Double-length key
KEYGENKY*	DES	Key generating key for DUKPT. Used with CSNBPTR, CSNBPTV, CSNBDKG, and CSNBUKD services	Double-length key
MAC*	AES	MAC generation and verification key	128-, 192-, or 256-bit key for AES
MAC	DES	MAC generation key	Single- or double-length key
MACVER	DES	MAC verification key	Single- or double-length key
NULL	AES, DES	Used to create a null CKDS entry	
OPINENC	DES	Output PIN encryption key	Double-length key
PINCALC*	AES	PIN calculation key	128-, 192-, or 256-bit key
PINGEN	DES	PIN generating key	Double-length key
PINPROT*	AES	PIN protection key	128-, 192-, or 256-bit key
PINPRW*	AES	PIN reference value key	128-, 192-, or 256-bit key
PINVER	DES	PIN verification key	Double-length key

All these types of keys are stored in the CKDS.

Note:

1. For compatibility with previous releases of CSF, KGUP stores internal versions of DATAM and DATAMV keys in the CKDS under the key types of MACD and MACVER, respectively.
2. Key types CIPHERXI, CIPHERXL, and CIPHERXO have control vectors with guaranteed unique key halves. Key-encrypting keys used to wrap these key types must have control vectors with guaranteed unique key halves. These key-encrypting keys can be generated using KGUP by specifying the DOULBEO keyword in the control statement.
3. The key types marked with an asterisk (*) require additional information to create the key. See the KEYUSAGE keyword for the values that must be specified.

ALGORITHM(DES|AES)

This keyword defines the algorithm of the key you are generating. DES is the default value except for key types not supported for the DES algorithm. When only one algorithm is supported for the key type, the keyword is optional. The supported algorithms for all key types is listed in the table under the TYPE keyword. Generated operational keys will be encrypted under the respective master key.

Note:

- To use an algorithm, the master key of the algorithm must be active.
- If you are going to create AES keys that use the variable-length format key token, the CKDS must be a variable-length record format CKDS and the key output data set must have a longer LRECL.

OUTTYPE (key-type)

This keyword specifies the type of complementary key you want KGUP to generate for export. This keyword is valid only when you are requesting KGUP to generate keys and you also specify the CLEAR or TRANSKEY keywords.

OUTTYPE is mutually exclusive with the KEY keyword.

Refer to Table 40 for a list of the default and optional complementary key types for each of the 11 different key types. If OUTTYPE is not specified, KGUP generates the default complementary key that is shown in this table.

Table 40. Default and Optional OUTTYPES Allowed for Each Key TYPE

Type	Algorithm	OUTTYPE (Default)	OUTTYPE (Allowed)
CIPHER	AES	CIPHER	CIPHER
CIPHER	DES	CIPHER	CIPHER, CIPHERXI, CIPHERXL, CIPHERXO, ENCIPHER, DECIPHER
CIPHERXI	DES	CIPHERXO	CIPHER, CIPHERXO, ENCIPHER
CIPHERXL	DES	CIPHERXL	CIPHER, CIPHERXL
CIPHERXO	DES	CIPHERXI	CIPHER, CIPHERXI, DECIPHER
CLRAES		Not Allowed	Not Allowed
CLRDES		Not Allowed	Not Allowed
DATA	AES	Not Allowed	Not Allowed
DATA	DES	DATA	DATA
DATAM	DES	DATAMV	DATAM, DATAMV
DATAMV	DES	Not Allowed	Not Allowed
DECIPHER	DES	ENCIPHER	CIPHER, CIPHERXO, ENCIPHER
DKYGENKY*	AES, DES	DKYGENKY*	DKYGENKY*
ENCIPHER	DES	DECIPHER	CIPHER, CIPHERXI, DECIPHER
EXPORTER	AES, DES	IMPORTER	IMPORTER
IMPORTER	AES, DES	EXPORTER	EXPORTER
IMPPKA	DES	EXPORTER	EXPORTER

Table 40. Default and Optional OUTTYPES Allowed for Each Key TYPE (continued)

Type	Algorithm	OUTTYPE (Default)	OUTTYPE (Allowed)
IPINENC	DES	OPINENC	OPINENC
KEYGENKY*	DES	KEYGENKY*	KEYGENKY*
MAC*	AES	MAC*	MAC*
MAC	DES	MACVER	MAC, MACVER
MACVER	DES	Not Allowed	Not Allowed
NULL	AES, DES	Not Allowed	Not Allowed
OPINENC	DES	IPINENC	IPINENC
PINCALC*	AES	Not Allowed	Not Allowed
PINGEN	DES	PINVER	PINVER
PINPROT*	AES	PINPROT*	PINPROT*, CIPHER
PINPRW*	AES	PINPRW*	PINPRW*
PINVER	DES	Not Allowed	Not Allowed

Note: The key types marked with an asterisk (*) require additional information to create the key and the key's complement. See the KEYUSAGE keyword for the values that must be specified.

TRANSKEY (key-label1[,key-label2])

This keyword identifies the label of a transport key that already exists in the CKDS. KGUP uses the transport key either to decrypt an imported key value or to encrypt a key value to send to another system. The algorithm of the transport key must match the key being wrapped, that is, an AES key must be wrapped with an AES transport key.

When KGUP generates a key, the program enciphers the key under the appropriate master key. KGUP may also generate a key value that can be used to create the key's complement. You can have KGUP encrypt the key value with a transport key. On the control statement, use the TRANSKEY keyword to specify an EXPORTER key-encrypting key that KGUP should use to encipher the complementary key. You can send the encrypted key value to another system to create the complementary key.

When you generate an importer key-encrypting key to encipher a key stored with data in a file, you can request that KGUP not generate the complementary export key-encrypting key. You do this by not specifying the TRANSKEY or CLEAR keyword. This is also true for CIPHER, DATA, and MAC keys.

For DES key types: When you input a key value that is in importable form, the key that is specified by the KEY keyword is enciphered under an IMPORTER key-encrypting key. KGUP reenciphers the key value from under the transport key to under a master key variant. On the control statement, you use the TRANSKEY keyword to specify the transport key that enciphers the key.

You can import or export a new version of a key that is encrypted under the current version of the same key. You can do this by specifying the same key label in the TRANSKEY keyword as in the LABEL or RANGE keyword on an UPDATE control statement.

Your site can generate keys for key exchange between two other sites. These sites do not need to know the clear value of the keys used for this communication. KGUP generates control statements that you send to the sites. Then the sites' KGUPs establish the keys they need for key exchange.

To do this procedure, submit an ADD or UPDATE control statement with two TRANSKEY key labels. The first TRANSKEY label identifies the transport key that is valid between your site and the first recipient site. The second TRANSKEY label identifies the transport key that is valid between your site and the second recipient site. KGUP generates of a pair of control statements to create the complementary pair of keys that are needed at the two sites.

Note: You cannot specify two DES NOCV key-encrypting keys. For more information about control vectors, see the description of the NOCV keyword.

The TRANSKEY keyword and the CLEAR keyword are mutually exclusive.

If you have specified a key type of NULL, CLRDES or CLRAES for the TYPE keyword, you cannot use the TRANSKEY keyword.

CLEAR

This keyword indicates that either:

- You are supplying an unencrypted key value with the KEY keyword.
- KGUP should create a control statement that generates an unencrypted complementary key value.

You can supply either encrypted or unencrypted key values to KGUP with the KEY keyword. On the control statement to supply the unencrypted key, you specify the CLEAR keyword.

When KGUP generates a key, KGUP enciphers the key under a master key variant. KGUP may also generate a key value to be used to create the key's complement. KGUP can create the complementary key value in unencrypted form. To generate an unencrypted complementary key value, you specify the CLEAR keyword. Your ICSF system must be in special secure mode to use this keyword.

The CLEAR keyword and the TRANSKEY keyword are mutually exclusive. You cannot use the CLEAR keyword on a control statement when you use the TRANSKEY keyword. You cannot use the CLEAR keyword if you specify a NULL, CLRDES or CLRAES key for the TYPE keyword.

NOCV

To exchange keys with systems that do not recognize CCA key tokens, ICSF provides a way to by-pass transport key variant processing. KGUP or an application program encrypts a key under the transport key itself not under the transport key variant. This is called NOCV processing.

The NOCV keyword indicates that the key that is generated or imported is a DES transport key to use in NOCV processing. The transport key has the NOCV flag set in the key control information when stored in the CKDS. The DOULBEO keyword is not valid with the NOCV keyword.

The NOCV keyword is only valid for generating transport keys. The keyword is not valid if you specify the TRANSKEY keyword with two transport key labels.

LENGTH(n), SINGLE and DOUBLEO

The LENGTH keyword specifies the length of the key value. Specifying the length of the key is optional. If the length is not specified, the default length will be used.

For AES keys and CLRAES, LENGTH(16) generates a 128-bit key, LENGTH(24) generates a 192-bit key, and LENGTH(32) generates a 256-bit key. The SINGLE and DOUBLEO keywords are not allowed.

For CLRDES keys, LENGTH(8) generates a single-length key, LENGTH(16) generates a double-length key and LENGTH(24) generates a triple-length key. The SINGLE and DOUBLEO keywords are not allowed.

For DES keys:

- LENGTH(8) generates a single-length key, LENGTH(16) generates a double-length key, and LENGTH(24) generates a triple-length key (DATA only).
- For most double-length key types, LENGTH(8) or SINGLE in an ADD or UPDATE statement causes KGUP to generate a double-length key with both key halves the same. On the KGUP panel, you can achieve this by specifying 8 in the LENGTH field for a double-length key type.
- For most double-length key types, specifying DOUBLEO causes KGUP to create a double length key with guaranteed unique key halves. The control vector is modified to indicate this. A key with this control vector cannot be used on systems with the Cryptographic Coprocessor Feature.

In any case, LENGTH is used only for generating keys. If you are specifying clear or encrypted key parts, do not use the LENGTH keyword (and do not fill in a value for LENGTH on the KGUP panel).

- The LENGTH keyword and the KEY keyword are mutually exclusive.
- The SINGLE and DOUBLEO keywords are mutually exclusive.
- The SINGLE and KEY keywords are mutually exclusive.
- The DOUBLEO keyword can be specified with the KEY keyword when two different key values are supplied. The control vector will be modified.

`KEY (key-value[,key-value[,key_value[,key_value]]])`

This keyword allows you to supply KGUP with a key value. KGUP can use this key value to add a key or update a key entry.

If you do not specify this keyword, KGUP generates the key value for you. You cannot use the RANGE keyword or the LENGTH keyword with this keyword. Each key part consists of exactly 16 characters that represent 8 hexadecimal values.

CAUTION:
KGUP does not create complementary key control statement for existing key labels, nor new a key label that has CLEAR parm specified in the KGUP statement.

This keyword is required when you specify either DATAMV, MACVER, or PINVER for the TYPE keyword. Because these type of keys require a complementary key to be used, you must always supply values for these types of keys.

This keyword is required when you specify CIPHERXI, CIPHERXO, DECIPHER, or ENCIPHER for the TYPE keyword and the TRANSKEY and OUTTYPE are not supplied. Because these type of keys require a complementary key to be used, you must always supply values for these types of keys.

For a double-length key, supply two key values. If you supply only one key value, KGUP will duplicate the key value as the second key value. KGUP concatenates these two identical values, and then stores and uses the key as if the key was double-length. For key types CIPHERXI, CIPHERXL, and CIPHERXO, the two keys values must be supplied and cannot be the same value.

For double-length keys, when you use the TRANSKEY keyword with the KEY keyword, the transport key you specify is the importer key that encrypts the key value. If you supply only one key value for a double-length key and also specify TRANSKEY, the TRANSKEY must be an NOCV importer.

For MAC and MACVER types, you can supply one or two key values.

For a DES DATA or CLRDES key, you can supply the key in one, two, or three parts.

For an AES DATA or CLRAES key, you must supply two, three or four parts.

For an AES CIPHER, EXPORTER or IMPORTER key, you must supply two, three or four parts. Note that when the TRANSKEY keyword is specified with these keys, KGUP will not create an entry in the Control Statement Output data set.

KEYUSAGE(key-usage-value1[,...,key-usage-value2])

This keyword defines key usage values for the key being generated. The usage values are used to restrict a key to a specific algorithm or usage.

The associated data for variable length tokens is described in Appendix B. of the Application Programmer's Guide. The DES control vector is described in Appendix C. of the Application Programmer's Guide.

The following values have been defined. The usage values are specific to a key type. The values can only be specified for the key type indicated in the tables below.

Note: Any value with a non-alphanumeric character must be enclosed in quotes when specified with the KEYUSAGE keyword. For example:

```
KEYUSAGE( 'CVVKEY-A' )
```

When a pair of keys is generated, one for the local system and the other for a remote system, both keys will be generated with the same key-usage flags when the KEYUSAGE keyword is used.

Table 41. Usage values for key types

Key type	Key algorithm	Key Usage Values
CIPHER	AES	The following values are optional: C-XLATE, V1PYLD and One or both may be specified: DECRYPT, ENCRYPT. **Note:** The key generated when KEYUSAGE is not specified will have only the DECRYPT and ENCRYPT key-usage. This is the default.
DKYGENKY	DES	One of the following must be specified: DKYL0, DKYL1, DKYL2, DKYL3, DKYL4, DKYL5, DKYL6, DKYL7 and One of the following must be specified: DALL, DDATA, DEXP, DIMP, DMAC, DMKEY, DMPIN, DMV, DPVR

Table 41. Usage values for key types (continued)

Key type	Key algorithm	Key Usage Values
DKYGENKY	AES	One of the following must be specified: D-PPROT, D-PCALC, D-PPRW and One of the following values must be specified: DKYL0, DKYL1, DKYL2 and The following values are required: KUF-MBE, DKYUSAGE
DKYGENKY	AES	One of the following must be specified: D-MAC, D-SECMSG and The following value is required: DKYUSAGE and One of the following values must be specified: KUF-MBE, KUF-MBP and One of the following values must be specified: DKYL0, DKYL1, DKYL2
DKYGENKY	AES	The following value is required: D-CIPHER and One of the following values must be specified: DKYL0, DKYL1, DKYL2 and The following value is optional: DKYUSAGE and One of the following values may be specified when DKYUSAGE is specified: KUF-MBE, KUF-MBP (KUP-MBE is the default)
DKYGENKY	AES	One of the following must be specified: D-ALL, D-EXP, D-IMP and One of the following values must be specified: DKYL0, DKYL1, DKYL2
EXPORTER	AES	The following value is optional: V1PYLD
IMPORTER	AES	The following value is optional: V1PYLD
KEYGENKY	DES	One of the following must be specified: UKPT, CLR8-ENC
MAC	DES	One of the following may be specified: ANY-MAC, CVVKEY-A, CVVKEY-B
MACVER	DES	One of the following may be specified: ANY-MAC, CVVKEY-A, CVVKEY-B

Table 41. Usage values for key types (continued)

Key type	Key algorithm	Key Usage Values
MAC	AES	One of the following must be specified: GENERATE, GENONLY, VERIFY and The following value must be specified: CMAC and One of the following is optional: DKPINOP, DKPINAD1, DKPINAD2 **Notes:** • One of either DKPINOP, DKPINAD1, or DKPINAD2 is required for keys to be used with the DK PIN services. • When DKPINOP, DKPINAD1, or DKPINAD2 is specified, GENERATE is not allowed.
PINCALC	AES	Three values must be specified: GENONLY, DKPINOP, and CBC.
PINPROT	AES	One of the following must be specified: ENCRYPT, DECRYPT and One of the following must be specified: DKPINOPP, DKPINOP, DKPINAD1 and The following value must be specified: CBC
PINPRW	AES	One of the following must be specified: GENONLY, VERIFY and The following values must be specified: DKPINOP, CMAC

Notes:

- **Diversified Key Generating Keys:** The key-derivation sequence level specifies the hierarchical level of the DKYGENKY. If the sequence level is non-zero, the DKYGENKY can only generate another DKYGENKY key with the sequence level decremented by one. If the sequence level is zero, the DKYGENKY can only generate the final diversified key (a non-DKYGENKY key) with the key type specified by the usage bits.
- **PINPROT Keys:** When specifying an AES CIPHER as the OUTTYPE for an AES PINPROT key, the key usage values must be ENCRYPT and DKINOPP. The key usage value for the AES CIIPHER key is DECRYPT.

Table 42. Meaning of usage values

Key Usage Value	Key types	Meaning
ANY-MAC	MAC, MACVER	The MAC usage field (control vector offset 0-3) is set to '0000'b. There is no restriction for this key. This is the default value.
C-XLATE	CIPHER	Restricts the key to be used with the cipher text translate2 service only.
CBC	PINCALC, PINPRW	Use the CBC encryption mode.

Table 42. Meaning of usage values (continued)

Key Usage Value	Key types	Meaning
CLR8-ENC	KEYGENKY	The CLR8-ENC key usage bit (control vector offset 19) is set to '1'b. The key may only be used with the 'CLR8-ENC' rule array keyword for CSNBDKG.
CMAC	MAC, PINPROT	Use the CMAC algorithm.
CVVKEY-A	MAC, MACVER	The MAC usage field (control vector offset 0-3) is set to '0010'b. When this key is used with CSNBCVG or CSNBCVV, it can only be used as the key A parameter. This is is valid with single- and double-length keys.
CVVKEY-B	MAC, MACVER	The MAC usage field (control vector offset 0-3) is set to '0011'b. When this key is used with CSNBCVG or CSNBCVV, it can only be used as the key B parameter. This is valid with single-length keys.
D-ALL	DKYGENKY	All key types may be derived except DKYGENKY keys.
D-CIPHER	DKYGENKY	CIPHER keys may be derived.
D-EXP	DKYGENKY	EXPORTER keys may be derived.
D-IMP	DKYGENKY	IMPORTER keys may be derived.
D-MAC	DKYGENKY	MAC keys may be derived.
D-PCALC	DKYGENKY	PINCALC keys may be derived.
D-PPROT	DKYGENKY	PINPROT keys may be derived.
D-PPRW	DKYGENKY	PINPRW keys may be derived.
D-SECMSG	DKYGENKY	SECMSG keys may be derived.
DALL	DKYGENKY	All key types may be generated except DKYGENKY and KEYGENKY keys. Usage is restricted by an access control point. See Diversified key generate callable service.
DDATA	DKYGENKY	Generate single- and double-length DATA keys
DECRYPT	PINPROT CIPHER	This key can be used to decrypt DK PIN blocks. This key can be used to decrypt data.
DEXP	DKYGENKY	Generate EXPORTER and OKEYXLAT keys
DIMP	DKYGENKY	Generate IMPORTER and IKEYXLAT keys
DKPINAD1	MAC, PINPROT	This key may be used in the DK PIN protection methods to create or verify a pin block to allow the changing of the account number associated with a PIN.
DKPINAD2	MAC	This key may be used in the DK PIN protection methods to create or verify an account change string to allow the changing of the account number associated with a PIN.
DKPINOP	MAC, PINCALC, PINPROT, PINPRW	This key may be used in the DK PIN protection methods as a general-purpose key. It may not be used as a special-purpose key.
DKPINOPP	PINPROT	This key is to be used to encrypt a PBF-1 format pin block for the specific purpose of creating a DK PIN mailer.

Table 42. Meaning of usage values (continued)

Key Usage Value	Key types	Meaning
DKYL0	DKYGENKY	Specifies that this key-generating key can be used to derive the key specified by the Key derivation and Derived key usage controls (AES) or control vector (DES).
DKYL1	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL0.
DKYL2	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL1.
DKYL3	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL2.
DKYL4	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL3.
DKYL5	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL4.
DKYL6	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL5.
DKYL7	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL6.
DKYUSAGE	DKYGENKY	Specifies that the DKYUSAGE keyword identifies key usage information for the key to be derived by the DKYGENKY. This value is required when the key type to be derived is MAC, PINCALC, PINPROT, PINPRW, and SECMSG. Not valid for D-ALL, D-CIPHER, D-IMP and D-EXP.
DMAC	DKYGENKY	Generate single- and double-length MAC keys
DMKEY	DKYGENKY	Generate secure messaging keys for encrypting keys
DMPIN	DKYGENKY	Generate secure messaging keys for encrypting PINs
DMV	DKYGENKY	Generate single- and double-length MACVER keys
DPVR	DKYGENKY	Generate PINVER keys
ENCRYPT	PINPROT CIPHER	This key can be used to encrypt DK PIN blocks. This key can be used to encrypt data.
GENERATE	MAC	This key can generate and verify MACs.
GENONLY	MAC, PINCALC, PINPRW	This key can be used to only generate data (MACs, PINs, or PRWs).
KUF-MBE	DKYGENKY	Specifies that the key usage fields of the key to be generated must be equal to the related generated key usage fields of the DKYGENKY generating key. Not valid for D-ALL, D-CIPHER, D-IMP and D-EXP.
KUF-MBP	DKYGENKY	Specifies that the key usage fields of the key to be generated must be permitted based on the related generated key usage fields of the DKYGENKY generating key. The key to be derived is not permitted to have a higher level of usage than the related key usage fields permit. The key to be derived is only permitted to have key usage that is less than or equal to the related key usage fields. Not valid for D-ALL, D-CIPHER, D-IMP and D-EXP.

Table 42. Meaning of usage values (continued)

Key Usage Value	Key types	Meaning
TRANSLAT	CIPHER	Restricts the key to be used with the cipher text translate2 service only.
UKPT	KEYGENKY	The UKPT key usage bit (control vector offset 18) is set to '1'b. The key may only be used in the CSNBPTR and CSNBPVR services.
VERIFY	MAC, PINPRW	This key can be used to verify data (MACs or PRWs).
V1PYLD	CIPHER, EXPORTER, IMPORTER	The generated key or keys will have version 1 (fixed-length) format of the payload for the variable-length symmetric key token. Applies to AES keys only.

Notes:

- **Diversified Key Generating Key Note:** The subtype field specifies the hierarchical level of the DKYGENKY. If the subtype is non-zero, then the DKYGENKY can only generate another DKYGENKY key with the hierarchy level decremented by one. If the subtype is zero, the DKYGENKY can only generate the final diversified key (a non-DKYGENKY key) with the key type specified by the usage bits.
- **PINPROT Keys:** When specifying an AES CIPHER as the OUTTYPE for an AES PINPROT key, the key usage values must be ENCRYPT and DKINOPP. The key usage value for the AES CIIPHER key is DECRYPT.
- **AES MAC Keys:** When DKPINOP, DKPINAD1, or DKPINAD2 is specified, GENERATE is not allowed.

Complementary key-usage values

When a pair of keys is generated, one for the local system and the other for a remote system,

- **For the AES CIPHER key type**, the key usage for the complementary key is determined from the values from the KEYUSAGE keyword as shown in Table 43. The other values do not have a complementary value and are copied.

Table 43. Complementary key-usage values for AES CIPHER

Key usage values	Complementary key usage values
ENCRYPT, DECRYPT	ENCRYPT, DECRYPT
ENCRYPT	DECRYPT
DECRYPT	ENCRYPT

- **For the AES MAC key type**, the key usage for the complementary key is determined from the values from the KEYUSAGE keyword as shown in Table 44. The other values do not have a complementary value and are copied. Note that for any key generated for the DK PIN methods, the local system gets the GENONLY key-usage. VERIFY key-usage is not allowed.

Table 44. Complementary key-usage values for AES MAC

Key usage values	Complementary key usage values
GENERATE	GENERATE

Table 44. Complementary key-usage values for AES MAC (continued)

Key usage values	Complementary key usage values
GENONLY	VERIFY
GENONLY, DKPINOP	VERIFY, DKPINOP
GENONLY, DKPINAD1	VERIFY, DKPINAD1
GENONLY, DKPINAD2	VERIFY, DKPINAD2
VERIFY	GENONLY

- **For the AES PINPROT key type**:
  - When TRANSKEY is specified, the ENCRYPT value is allowed for the local system and DECRYPT values is allowed for the remote system.
  - When CLEAR is specified, ENCRYPT and DECRYPT are complementary values.
  - The other values do not have a complementary value and are copied.
- **For the AES PINPRW key types**:
  - When TRANSKEY is specified, the GENONLY value is allowed for the local system and VERIFY values is allowed for the remote system.
  - When CLEAR is specified, GENONLY and VERIFY are complementary values.
  - The other values do not have a complementary value and are copied.
- **For the AES DKYGENKY key type**, the key usage values for the complementary key are the complement of the generated key. There are restrictions for the values specified in the DKYGENKYUSAGE keyword. See the DKYGENKYUSAGE keyword description.
- **For all other key types**, both keys are generated with the same key-usage values.

DES

This keyword is no longer supported but is tolerated.

DKYGENKYUSAGE(key-usage-value1[,...,key-usage-value2])

This keyword defines key usage values to be supplied for the AES DKYGENKY key being generated. This keyword is required when the DKYUSAGE value is specified in the KEYUSAGE keyword.

The following values have been defined. The usage values are specific to the key type to be derived. The values can only be specified for the key type indicated in Table 45 and Table 46 on page 206. The values for the specific key types are detailed in this document in the Key Token Build2 callable service description.

Note: Any value with a non-alphanumeric character must be enclosed in quotes when specified with the DKYGENKYUSAGE keyword. For example: DKYGENKYUSAGE(’CVVKEY-A’).

Table 45. Values by type for DKYGENKYUSAGE

Type of key to be derived	DKYGENKYUSAGE values
CIPHER	The following values are optional: C-XLATE, DECRYPT, ENCRYPT **Note:** The key generated when DKYGENKYUSAGE is not specified will have DECRYPT and ENCRYPT key-usage. This is the default.

Table 45. Values by type for DKYGENKYUSAGE (continued)

Type of key to be derived	DKYGENKYUSAGE values
MAC	One of the following values is required: GENERATE, GENONLY, VERIFY and The following value is required: CMAC and One of the following values is optional: DKPINAD1, DKPINAD2, DKPINOP **Notes:** • One of DKPINOP, DKPINAD1, or DKPINAD2 is required for keys to be used with the DK PIN services. • When DKPINOP, DKPINAD1, or DKPINAD2 is specified, GENERATE is not allowed.
PINCALC	The following values are required: GENONLY, CBC, DKPINOP.
PINPROT	One of the following values is required: DECRYPT, ENCRYPT and The following value is required: CBC and One of the following values is required: DKPINAD1, DKPINOP, DKPINOPP
PINPRW	One of the following values is required: GENONLY, VERIFY and The following values are required: CMAC, DKPINOP
SECMSG	The following value is required: SMPIN and One of the following values is required: ANY-USE, DPC-ONLY

Table 46. Meaning of usage values

Value	Key types	Description
ANY-USE	SECMSG	The use of the key in a callable service is not restricted.
CBC	PINPROT, PINCALC	The derived key must use the CBC encryption mode.
CMAC	MAC, PINPRW	The derived key must use the CMAC algorithm.
C-XLATE	CIPHER	Restricts the key to be used with the cipher text translate2 service only.
DPC-ONLY	SECMSG	The use of the key is restricted to the DK PIN Change service.
DECRYPT	CIPHER, PINPROT	The derived key may be used to decrypt PIN blocks.
DKPINAD1	MAC, PINPROT	The derived key may be used to create or verify a pin block to allow changing the account number associate with a PIN for the DK PIN methods.

Table 46. Meaning of usage values (continued)

Value	Key types	Description
DKPINAD2	MAC	The derived key may be used to create or verify an account change string to allow changing the account number associated with a PIN for the DK PIN methods.
DKPINOP	MAC, PINCALC, PINPROT, PINPRW	The derived key may be used as a general purpose key for the DK PIN methods.
DKPINOPP	PINPROT	The derived key may be used to encrypt a PIN block for the specific purpose of creating a PIN mailer for the DK PIN methods.
ENCRYPT	CIPHER, PINPROT	The derived key may be used to encrypt PIN blocks.
GENERATE	MAC	The derived key may be used to generate and verify MACs.
GENONLY	MAC, PINCALC	The derived key may be used to generate MACs or PINs.
SMPIN	SECMSG	Enable the encryption of PINs in an EMV secure message.
VERIFY	MAC	The derived key may be used to verify MACs.

Complementary DKYGENKY usage values

When a pair of DKYGENKY keys is generated, one for the local system and the other for a remote system, the complementary key will have a different value as shown in Table 47. Values that do not appear in the table are copied for the complementary key.

Table 47. Complementary values for usage values

Type of key to be derived	DKYGENKY usage value	Complementary value
CIPHER	ENCRYPT	DECRYPT
CIPHER	DECRYPT	ENCRYPT
MAC	GENERATE	GENERATE
MAC	GENONLY	VERIFY
MAC	VERIFY	GENONLY
MAC with DKPINOP, DKPINAD1 or DKPINAD2	GENONLY	VERIFY
PINCALC	Not allowed	Not allowed
PINPROT	ENCRYPT	DECRYPT
PINPRW	GENONLY	VERIFY

Attention: NOCV processing takes place automatically when KGUP or an application specifies the use of a transport key that was generated by KGUP with a NOCV keyword specified.

The use of NOCV processing eliminates the ability of the system that generates the key to determine the use of the key on a receiving system. Therefore, access to these keys should be strictly controlled. For a description of security considerations, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Using the ADD and UPDATE control statements for key management and distribution functions

You use the ADD and UPDATE control statements to run KGUP for functions that involve key generation, maintenance, and distribution. For ADD and UPDATE control statements, KGUP either imports a key value that you supply or generates a key value. KGUP allows the creation and maintenance of clear key tokens in the CKDS. This topic describes the combinations of control statement keywords you use to perform these functions. Table 48 shows the keyword combinations permitted on ADD and UPDATE control statements.

Table 48. Keyword Combinations Permitted in ADD and UPDATE Control Statements for DES Keys. Keyword Combinations Permitted in ADD and UPDATE Control Statements for DES Keys

Control Statement	LABEL or RANGE	TYPE	OUTTYPE	TRANSKEY or CLEAR	NOCV	ALGORITHM	LENGTH or KEY
ADD	Yes	Yes	Yes[1]	Yes[2]	Yes[3]	Yes	Yes[1]
UPDATE	Yes	Yes	Yes[1]	Yes[2]	Yes[3]	Yes	Yes[1]

Note:
1. OUTTYPE can be used with either TRANSKEY or CLEAR but is mutually exclusive with KEY.
2. TRANSKEY is not valid when TYPE is NULL, CLRDES or CLRAES.
3. NOCV is not valid when TRANSKEY is specified with two key labels. It is not valid when TYPE is CLRDES or CLRAES.

Table 49. Keyword Combinations Permitted in ADD and UPDATE Control Statements for AES Keys. Keyword Combinations Permitted in ADD and UPDATE Control Statements for AES Keys

Control Statement	LABEL or RANGE	TYPE	OUTTYPE	TRANSKEY or CLEAR	ALGORITHM	LENGTH or KEY
ADD	Yes	Yes	Yes[1]	Yes[2]	Required	Yes[1]
UPDATE	Yes	Yes	Yes[1]	Yes[2]	Required	Yes[1]

Note:
1. OUTTYPE can be used with either TRANSKEY or CLEAR but is mutually exclusive with KEY.
2. TRANSKEY is not valid when TYPE is NULL, CLRDES or CLRAES.

To Import Keys

You use an ADD or UPDATE control statement to supply a value to KGUP. The program receives the value, enciphers the value under a master key variant, and places the value in a CKDS entry. The value that you supply may be in clear form or it may be encrypted under a transport key. The statement that contains the value may be sent from another system. The other system sends the value to create a key on your system. This key is the complement of a key that was generated on the other system.

You can supply a transport key value to KGUP from a system that does not use control vectors. You use the key for key exchange with that system. KGUP places the key into the CKDS with an indication that the key is to be used without control vectors.

Import a Clear Key Value: You can supply a clear key value on a control statement for KGUP to import.

These statements show the syntax when you supply a clear key value to KGUP.

Note: For these control statements, your system should be in special secure mode.

When you supply a single-length, clear key value:

```
ADD or UPDATE  LABEL(label)    TYPE(data,exporter,importer,
mac,macver, or any PIN key)  CLEAR  KEY(key-value)
```

When you supply a double-length, clear key value:

```
ADD or UPDATE  LABEL(label)  TYPE(data,datam,datamv,exporter,importer,
or any PIN key)  CLEAR  KEY(key-value, key-value)
```

When you supply a triple-length, clear key value:

```
ADD or UPDATE  LABEL(label)  TYPE(data),
CLEAR  KEY(key-value, key-value, key-value)
```

When you supply a single-length clear key value and you use the key to exchange keys with a cryptographic product that does not use control vectors or double-length keys:

```
ADD or UPDATE  LABEL(label)  TYPE(exporter or importer),
CLEAR  KEY(key-value)  NOCV
```

When you supply a double-length, clear key value, and you use the key to exchange keys with a cryptographic product that does not use control vectors:

```
ADD or UPDATE  LABEL(label)  TYPE(exporter or importer),
CLEAR  KEY(key-value,ikey-value)  NOCV
```

Import a Clear Key Value for AES keys: You can supply a clear key value on a control statement for KGUP to import.

When you supply a 128-bit clear key value for an AES DATA key:

```
ADD or UPDATE  LABEL(label)  TYPE(data) ALGORITHM(AES),
CLEAR KEY(key-value,key-value)
```

When you supply a 192-bit clear key value for an AES key that uses the variable-length token:

```
 ADD or UPDATE LABEL(label) TYPE(cipher, exporter, or importer),
 ALGORITHM(AES) CLEAR KEY(key-value,key-value)
```

CLRDES and CLRAES key types: The CLEAR keyword is not allowed because the key type indicates that the KEY is a clear key value. Also, special secure mode is not required for these key types.

```
ADD or UPDATE LABEL(label) TYPE(clraes),
KEY(key-value, key-value)
ADD or UPDATE LABEL(label) TYPE(clrdes),
KEY(key-value, key-value)
```

Import an Encrypted Key Value for DES keys: When you supply KGUP with an encrypted key value, the value is encrypted under a transport key. The transport key is one key in a complementary key pair that you share with another system. When the other system's KGUP generated a key, the program also stored a control statement to use to create the complementary key. The other system sends the control statement to your system. You can use the statement to supply an encrypted key value to KGUP to create the key.

These statements show the syntax when you supply an encrypted key value to KGUP.

When you supply a single-length, encrypted key value:

```
ADD or UPDATE  LABEL(label)  TYPE(data,exporter,importer,
mac,macver, or any PIN key) TRANSKEY(key-label 1)  KEY(key-value)
```

When you supply a double-length, encrypted key value:

```
ADD or UPDATE  LABEL(label)  TYPE(data,datam,datamv,exporter,importer,
or any PIN key) TRANSKEY(key-label 1)  KEY(key-value,ikey-value)
```

When you supply a triple-length, encrypted key value:

```
ADD or UPDATE  LABEL(label)  TYPE(data),
TRANSKEY(key-label 1)  KEY(key-value, key-value, key-value)
```

When you supply a single-length, encrypted key value, and you are using the key to exchange keys with a cryptographic product that does not use control vectors or double-length keys:

```
ADD or UPDATE  LABEL(label)  TYPE(exporter or importer),
TRANSKEY(key-label 1)  KEY(key-value)  NOCV
```

Note: Single-length keys with replicated key parts can be brought in under a TRANSKEY only if the TRANSKEY is an NOCV IMPORTER.

When you supply a double-length encrypted key value and you will use the key to exchange keys with a cryptographic product that does not use control vectors:

```
ADD or UPDATE  LABEL(label)  TYPE(exporter or importer),
TRANSKEY(key-label 1)  KEY(key-value,ikey-value)  NOCV
```

To Generate Keys

You use an ADD or UPDATE control statement to have KGUP generate a key value to place in the CKDS. The program generates the value, enciphers the value under a master key variant, and places the value in the CKDS. When KGUP generates a key, the program may also store information to create the key's complement in a data set.

You can have KGUP generate a transport key that you use to send or receive keys from a system that does not use control vectors. KGUP places the key into the CKDS with an indication that the key is to be used without control vectors.

Generate an Importer Key For File Encryption: You can have KGUP create an importer key without having KGUP store information about the complement of the key. You do not use the importer key in key exchange with another system. You use the importer key to encrypt a data-encrypting key that you use to encrypt data in a file on your system. You can store the data-encrypting key with the file, because the data-encrypting key is encrypted under the importer key.

These statements show the syntax when you generate an importer key to use in file encryption on a system:

When you generate a single-length key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(importer)  SINGLE
```

When you generate a double-length key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(importer)
```

Generate an AES data key: You can have KGUP create an AES data key. The keys may be 128-, 192- or 256-bits in length.

These statements show the syntax when you generate an AES data key on a system.

When you generate a 128-bit key value:

```
ADD or UPDATE  ALGORITHM(AES) LABEL(label) or RANGE(start-label,end-label),
TYPE(data)
```

When you generate a 192-bit key value:

```
ADD or UPDATE  ALGORITHM(AES) LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(24)
```

Generate a Complementary, Clear Key Value: You can have KGUP store complementary key information when KGUP generates a key. This information includes the key value. You send the information to another system which uses the information to generate the complementary key. KGUP stores the key value to create the complementary key in either clear or encrypted form. KGUP stores information both in and not in the form of a control statement.

These statements show the syntax when you have KGUP store the complementary key value in clear form.

Note: For these control statements, your system should be in special secure mode.

When you generate a single-length, transport or PIN clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter,importer,ipinenc,opinenc, or pingen) CLEAR  SINGLE
```

When you generate a single-length, DATA clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(8) CLEAR
```

When you generate a double-length, DATA clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(16) CLEAR
```

When you generate a triple-length, DATA clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(24) CLEAR
```

When you generate a single-length, MAC clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(mac) OUTTYPE(mac or macver) CLEAR
```

When you generate a double-length, DATAM clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(datam) LENGTH(16) OUTTYPE(datam or datamv) CLEAR
```

When you generate a single-length, PINGEN clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(pingen) LENGTH(8) CLEAR
```

When you generate a double-length, clear key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter,importer,ipinenc,opinenc, or pingen) CLEAR
```

When you generate a single-length, clear key value, and you are using the key to exchange keys with a cryptographic product that does not use control vectors:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter or importer)  CLEAR  NOCV  SINGLE
```

When you generate a double-length, clear key value, and you are using the key to exchange keys with a cryptographic product that does not use control vectors:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(16) CLEAR NOCV
```

When you generate a triple-length, clear key value, and you are using the key to exchange keys with a cryptographic product that does not use control vectors:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(24)  CLEAR  NOCV
```

When you generate a double-length, clear key value, and you are using the key to exchange keys with a cryptographic product that does not use control vectors:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter or importer)  CLEAR  NOCV
```

When you generate a clear key value to transport data-encrypting keys for use in the DES algorithm:

```
ADD or UPDATE LABEL(label) TYPE(exporter or importer) CLEAR
```

Generate a Complementary, Encrypted Key Value: KGUP encrypts the complementary key value under the exporter key that you specify.

These statements show the syntax when you have KGUP generate the complementary key value in encrypted form.

When you generate a single-length, transport or PIN encrypted key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter,importer,ipinenc,opinenc, or pingen),
TRANSKEY(key-label 1)  SINGLE
```

When you generate a single-length, DATA encrypted key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) OUTTYPE(data) TRANSKEY(key-label 1)
```

When you generate a single-length, MAC encrypted key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(mac) OUTTYPE(mac or macver) TRANSKEY(key-label 1)
```

When you generate a double-length, encrypted key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter,importer,ipinenc,opinenc, or pingen)  TRANSKEY(key-label 1)
```

When you generate a double-length DATA encrypted key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data or datam)  LENGTH(16) TRANSKEY(key-label 1)
```

When you generate a double-length DATAM encrypted key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(datam)  TRANSKEY(key-label 1)
```

When you generate a triple-length DATA encrypted key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data)  LENGTH(24) TRANSKEY(key-label 1)
```

When you generate a single-length, encrypted key value, and you are using the key to exchange keys with a cryptographic product that does not use control vectors:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter or importer)  TRANSKEY(key-label 1)  SINGLE  NOCV
```

When you generate a double-length, encrypted key value, and you are using the key to exchange keys with a cryptographic product that does not use control vectors.

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter or importer)  TRANSKEY(key-label 1)  NOCV
```

Generate a Complementary Key Pair For Other Systems: You can also use KGUP as a key distribution center. KGUP generates a pair of complementary key values that are both used on other systems. KGUP encrypts the values under appropriate variants of two different exporter key-encrypting keys. KGUP does not alter your system's CKDS. The program stores two control statements each containing one of the keys that are encrypted under a transport key. You send the statements to two other sites which can create the keys and use the keys to exchange keys.

These statements show the syntax when you have KGUP generate a pair of complementary key values to send to other systems.

When you generate single-length transport or PIN key values:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter,importer,ipinenc,opinenc, or pingen),
TRANSKEY(key-label 1,key-label 2)  SINGLE
```

When you generate single-length DATA key values:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) OUTTYPE(data) TRANSKEY(key-label 1,key-label 2)
```

When you generate double-length DATA key values:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(16) TRANSKEY(key-label 1,key-label 2)
```

When you generate triple-length DATA key values:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(data) LENGTH(24) TRANSKEY(key-label 1,key-label 2)
```

When you generate single-length MAC key values:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(mac) OUTTYPE(mac or macver) TRANSKEY(key-label 1,key-label 2)
```

When you generate double-length DATAM key values:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label)
TYPE(datam) OUTTYPE(datam or datamv),
TRANSKEY(key-label 1,key-label 2)
```

When you generate a double-length key value:

```
ADD or UPDATE  LABEL(label) or RANGE(start-label,end-label),
TYPE(exporter,importer,ipinenc,opinenc, or pingen),
TRANSKEY(key-label 1,key-label2)
```

To Create NULL Keys

You can use KGUP to create an initial record in the CKDS. To do this, you create an ADD control statement with a key TYPE of NULL. Once you have created this key record, you can use the Key Record Write callable service to place a key value in the record.

If you are generating a large number of keys, you will get better performance if you create the NULL key records with KGUP. This is preferrable to using the Key_Record_Create callable service.

Create NULL Key Records: You can use KGUP to create a single NULL key record or a range of NULL key records. This statement shows the syntax you use:

```
ADD  LABEL(label) or RANGE(start-label,end-label)  TYPE(null)
```

Syntax of the RENAME Control Statement

The RENAME control statement changes the label of a key entry in the CKDS. KGUP does not change any other information in the entry.

The RENAME control statement has this syntax:

RENAME

LABEL(*old-label,new-label*)

TYPE(*key-type*)

Figure 52. RENAME Control Statement Syntax

LABEL(old-label,new-label)
This keyword specifies the labels of the CKDS entries that you want KGUP to process. For the general rules about key label conventions and uniqueness, see "General Rules for CKDS Records" on page 190.

First you specify the old label which is the current label in the CKDS that KGUP changes. Then you specify the new label to replace the old label.

TYPE(key-type)
Because you can use the same label in entries with different key types, this keyword specifies the type of key for the old entry and the new entry.

Syntax of the DELETE Control Statement

DELETE control statements instruct KGUP to remove key entries from the CKDS.

The DELETE control statement has this syntax:

DELETE

{LABEL(*label1[,...,label64]*) | RANGE(*start-label,end-label*)}

TYPE(*key-type*)

Figure 53. DELETE Control Statement Syntax

LABEL (label1[,...,label64])

This keyword defines the names of the key entries for KGUP to delete from the CKDS. KGUP deletes a separate entry for each label.

You must specify at least one key label, and you can specify up to 64 labels with the LABEL keyword. For the general rules about key label conventions and uniqueness, see "General Rules for CKDS Records" on page 190.

On a KGUP control statement, you must specify either the LABEL or RANGE keyword.

RANGE (start-label, end-label)

This keyword defines the range of the multiple labels that you want KGUP to delete from the CKDS.

The label consists of between 2 and 64 characters that are divided as follows:

- The first 1 to 63 characters are the label base. These characters must be identical on both the start-label and end-label and are repeated for each label in the range. For the general rules about key label conventions and uniqueness, see "General Rules for CKDS Records" on page 190.
- The last 1 to 4 characters form the suffix. The number of digits in the start-label and end-label must be the same, and the characters must all be numeric. These numeric characters establish the range of labels KGUP creates. The start-label numeric value must be less than the end-label numeric value.

TYPE(key-type)

Because you can use the same label in entries with different key types, this keyword specifies the type of key that is being deleted.

To Delete Keys

You can use a KGUP control statement to remove a key or a range of keys from the CKDS. This statement shows the syntax when you delete keys from the CKDS:

```
DELETE  LABEL(label) or RANGE(start-label,end-label)
TYPE(data,exporter,importer,ipinenc,mac,macver,
null,opinenc,pingen, or pinver)
```

Syntax of the SET Control Statement

The SET control statement specifies data you want KGUP to pass to the installation-defined exit routine for processing.

The SET control statement has this syntax:

SET

INSTDATA(*data-value*)

Figure 54. SET Control Statement Syntax

INSTDATA(data-value)

This keyword specifies the data KGUP sends to the KGUP exit routine while processing control statements.

During a KGUP job, the data you specify with the INSTDATA keyword is held and sent to the exit routine each time the exit is entered for control statement processing. The same information is sent until KGUP encounters another SET control statement. The data you specified in this SET control statement replaces the data you specified in the previous SET control statement.

A KGUP exit routine performs different operations that depend on the data that is sent and the time of the call. A KGUP exit routine can change the data you send the exit and send the changed data to the user area of a key entry in the CKDS. The user area of a key entry can contain any information that you choose to store in the area.

For more information about the KGUP exit routine, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

The maximum length of the character string that you can specify to an exit routine is 52 bytes. If you use blanks or special characters within the string, then you must delimit the entire string with single quotes ('). These quotes are not included as part of the 52-byte string.

Syntax of the OPKYLOAD Control Statement

The OPKYLOAD control statement specifies the operational key created by the TKE workstation on a CCA coprocessor that you want KGUP to load to the CKDS. An SMF record type 82 subtype 7 will be generated when the key is written to the CKDS.

The OPKYLOAD control statement has this syntax:

```
OPKYLOAD

   LABEL (key-label)
   SERNBR (coprocessor-serial-number)
   [NOCV]
```

Figure 55. OPKYLOAD Control Statement Syntax

LABEL (*key-label*)
This label must match the label used to create the key by the TKE workstation on the coprocessor.

SERNBR (*coprocessor-serial-number*)
The serial number is available on the Service Element panels and the ICSF coprocessor management panel. The coprocessor-serial-number is the serial number of the coprocessor where the key identified by the key-label has been loaded from the TKE workstation.

NOCV
NOCV specifies that the DES IMPORTER/EXPORTER key being written to the CKDS should be NOCV IMPORTER/EXPORTER. The key must have a default control vector.

Examples of Control Statements

Example 1: ADD Control Statement

This example shows a control statement that specifies that KGUP add an entry to the CKDS.

```
ADD  TYPE(IMPORTER) LABEL(DASDOCT93401E)
```

KGUP checks that an entry labeled DASDOCT93401E with a keytype of importer does not already exist in the CKDS. KGUP allows EXPORTER, IMPORTER, IPINENC, PINGEN, PINVER, and OPINENC key pairs to have the same label. All other key types require a unique label. If the key entry already exists, KGUP stops processing the control statement.

If the entry does not exist, KGUP creates the entry with a label of DASDOCT93401E and type of IMPORTER. KGUP generates a double-length key and encrypts the key under the master key. KGUP places the key in the entry.

Note: Because neither the TRANSKEY nor CLEAR keyword is specified, KGUP does not create a complementary key. You cannot use this key to communicate with another system. You can, however, use the key to encipher a key stored with data in a file. CIPHER, CIPHERXL, DATA, DATAM, DKYGENKY, IMPORTER, KEYGENKY and MAC are the only key types that do not require either the TRANSKEY or CLEAR keyword specified.

Example 2: ADD Control Statement with CLEAR Keyword

This example shows a control statement that specifies that KGUP add an entry to the CKDS. Because the CLEAR keyword is specified, KGUP processes only this control statement if ICSF is in special secure mode.

```
ADD  TYPE(EXPORTER)  LABEL(ATMBRANCH5MO001)  CLEAR
```

KGUP checks that an entry with the label ATMBRANCH5MO001 with the type EXPORTER does not already exist in the CKDS. KGUP allows EXPORTER, IMPORTER, IPINENC, PINGEN, PINVER, and OPINENC key pairs to have the same label. All other key types require a unique label. If the entry already exists, KGUP stops processing the control statement.

If the entry does not exist, KGUP creates the entry for the label specified and the type exporter. KGUP generates a double-length key, encrypts the key under the master key, and places the key in the entry.

KGUP stores information to the key output data set. You can send the information to another system that does not use KGUP. The other system uses the information to create the complements of the keys you created. The information contains the clear key value and specifies the key type as importer.

For example, the control statement would be in this format:

```
ADD  TYPE(IMPORTER)  LABEL(ATMBRANCH5MO001)  CLEAR,
KEY(6709E5593933DA00,9099937DDE93A944)
```

The key value is the clear key value of the key created. The type of key is the complement of the type of key created.

Note: The key in the previous example is a mixed parity key. KGUP imports mixed parity keys, but issues a warning message.

Example 3: ADD Control Statement with one TRANSKEY Keyword

This example shows a control statement that specifies that KGUP add an entry to the CKDS. Because the TRANSKEY keyword is specified, KGUP also creates a control statement that another installation uses to create the complement of the key for PIN exchange.

```
ADD  TYPE(IPINENC)  LABEL(LOCTOJWL.JULY03)  TRANSKEY(SENDJWL.JULY03)
```

KGUP checks that an entry with the label LOCTOJWL.JULY03 for an input PIN-encrypting key does not already exist in the CKDS.KGUP allows EXPORTER, IMPORTER, IPINENC, PINGEN, PINVER, and OPINENC key pairs to have the same label. All other key types require a unique label. If the entry already exists, KGUP stops processing the control statement.

If the entry does not exist, KGUP creates the entry with a label of LOCTOJWL.JULY03 and type of IPINENC. KGUP generates a double-length key. KGUP encrypts the key under the master key and places the key in the entry.

KGUP stores information to the key output data set. You can send the information to another system that does not use KGUP. The other system uses the information to create the complement of the key you created. The information contains the key in exportable form. The key is encrypted under the exporter key, labelled SENDJWL.JULY03, that was specified by the TRANSKEY keyword. The information specifies the key type as output PIN-encrypting key (OPINENC).

Note: If SENDJWL.JULY03 is a DES NOCV exporter, the exportable OPINENC key is encrypted without a control vector.

KGUP stores a control statement to the control statement output data set. You can send the control statement to another system. The other system's KGUP uses the statement to create a key that complements the key that you created.

For example, the control statement would be in this format:

```
ADD  TYPE(OPINENC)  LABEL(LOCTOJWL.JULY03)  TRANSKEY(SENDJWL.JULY03),
KEY(6709E5593933DA00,9099937DDE93A944)
```

The key value is the encrypted value of the key that KGUP created. The key is encrypted under the exporter key, labeled SENDJWL.JULY03, which was the transport key label that was specified on the original control statement. The type of key is the complement of the type of key it created.

Example 4: ADD Control Statement with two TRANSKEY Keywords

This example shows a control statement specifying that KGUP create keys for key exchange between two other sites.

```
ADD  TYPE(EXPORTER)  LABEL(JWL@SSIJULY03),
TRANSKEY(SENDTOJWLJULY03,SENDTOSIIJULY03)
```

KGUP generates a key value and encrypts the value under the variants of the exporter key-encrypting keys that are specified by the TRANSKEY keyword. KGUP does not alter the CKDS in any way.

KGUP stores these two control statements to the control statement output data set:

```
ADD  TYPE(EXPORTER)  LABEL(JWL@SSIJULY03)  TRANSKEY(SENDTOJWLJULY03),
KEY(4542E37B570033AD,3C00F6850A99E11B)

ADD  TYPE(IMPORTER)  LABEL(JWL@SSIJULY03)  TRANSKEY(SENDTOSIIJULY03),
KEY(6709E5993933DA00,1449A3D9ED0A1586)
```

The control statements create keys that complement each other. You send the statements to two sites that want to exchange keys. The receiving sites process the statements to create a complementary pair of transport keys.

KGUP also stores information to create the keys in the key output data set.

Example 5: ADD Control Statement with a Range of NULL Keys

This example shows a control statement that creates a range of empty key records in a CKDS. Once the key labels exist, you can enter key types and key values for these records in several ways. One method is to use KGUP to create UPDATE control statements. Another method is to write application programs that use the Key_Record_Write callable service to add key types and key values to the existing empty key records.

```
ADD  TYPE(NULL)  RANGE(BRANCH5M0001,BRANCH5M0025)
```

KGUP checks for any entries with labels between `BRANCH5M001` and `BRANCH5M0025` in the CKDS. If any entries in this range already exist, KGUP processes the control statement up to the point where a duplicate label is found. It then stops processing the control statement and issues error messages.

If no entries exist, KGUP creates a range of 25 sequentially-numbered key records and adds them to the CKDS.

Example 6: ADD Control Statement with OUTTYPE and TRANSKEY Keywords

This example shows a control statement that specifies that KGUP add an entry with the key type of DATAM to the CKDS. The TRANSKEY keyword instructs KGUP to create a control statement for an intermediate node to use to create the complement DATAMV key for intermediate node data translation.

```
ADD  LABEL(DATAKEY.TO.TRANSLATION)  TYPE(DATAM)  OUTTYPE(DATAMV),
TRANSKEY(TKBRANCH2.INTER)
```

KGUP checks that an entry with the label `DATAKEY.TO.TRANSLATION` does not already exist in the CKDS, because DATAM keys require unique labels. If the entry already exists, KGUP stops processing the control statement.

If the entry does not exist, KGUP creates the entry with a label of `DATAKEY.TO.TRANSLATION` and a type of DATAM. KGUP then generates a single-length key, encrypts the key under the master key variant for a DATAM key, and places the key in the CKDS entry.

KGUP stores information to the key output data set. You can send the information to another system that does not use KGUP. The other system uses the information to create the complement of the key you created. The information contains the key value of the key in exportable form. The key is encrypted under the exporter key, labeled `TKBRANCH2.INTER`, that was specified by the TRANSKEY keyword. The information specifies the key type as data-translation key (DATAMV).

KGUP stores a control statement to the control statement output data set. You can send the control statement to another system. The other system's KGUP uses the statement to create a key that complements the key you created.

For example, the control statement would be in this format:

```
ADD  TYPE(DATAMV)  LABEL(DATAKEY.TO.TRANSLATION),
TRANSKEY(TKBRANCH2.INTER) KEY(2509F2869257BD00,1616161616161616)
```

The key value is the encrypted value of the key that KGUP created. The key is encrypted under the exporter key, labelled `TKBRANCH2.INTER`, which was the transport key label that was specified on the original control statement. The type of key is the complement of the type of key it created.

Example 7: UPDATE Control Statement with Key Value and Transkey Keywords

This example shows a control statement that specifies that KGUP import a key value. KGUP places the key value into an entry in the CKDS that already exists.

```
UPDATE  LABEL(PINVBRANCH5M0002)  TYPE(PINVER)  TRANSKEY(TKBRANCH5JUNE99),
KEY(7165865940460A48,2237451B4545718B)
```

The key value on the control statement is encrypted under a transport key that is shared with another system. The label for the transport key is `TKBRANCH5JUNE99`. KGUP uses the importer key labelled `TKBRANCH5JUNE99` to decrypt the key value.

KGUP encrypts the key value under the master key variant for a PIN verification key. KGUP then places the key in a key entry labelled `PINVBRANCH5M0002` with the type PINVER in the CKDS.

Example 8: DELETE Control Statement

This example shows a control statement that specifies that KGUP delete an entry from the CKDS.

```
DELETE  LABEL(GENBRANCH2M0003)  TYPE(PINGEN)
```

KGUP deletes the entry with a label of `GENBRANCH2M0003` and type of PIN generation key from the CKDS. If KGUP cannot find the entry, KGUP gives you an error message.

Example 9: RENAME Control Statement

This example shows a control statement that specifies that KGUP rename an entry in the CKDS.

```
RENAME  LABEL(JWL@SSIDEC97,JWL@SSIJUNE99)  TYPE(EXPORTER)
```

KGUP checks if an entry with a label of `JWL@SSIJUNE99` and a key type of EXPORTER already exists in the CKDS. If the entry does exist, KGUP does not process the control statement. KGUP checks if an entry with the label `JWL@SSIDEC97` contains a key type of EXPORTER exists. If the entry exists, KGUP renames the entry `JWL@SSIJUNE99`.

Example 10: SET Control Statement

This example shows a control statement that specifies that KGUP send certain installation data every time an exit is called during KGUP processing. KGUP sends the data every time an exit is called until KGUP encounters another SET statement or the job stream completes.

```
SET  INSTDATA('This key is valid effective 9/9/99')
```

KGUP sends the installation data each time an installation exit is called during KGUP processing.

Example 11: OPKYLOAD Control Statement

This example shows a control statement to load a key into the CKDS from any CCA cryptographic coprocessor. The serial number of the card is 94000011. A key has been loaded on the card with the label ERC033.DEC50.

```
OPKYLOAD LABEL(ERC033.DEC50) SERNBR(94000011)
```

KGUP checks the CKDS for the label and will fail if the label exists. KGUP then queries the coprocessor to see if the key exists on the card. If the key exists, the key token is retrieved from the card and loaded into the CKDS.

Example 12: OPKYLOAD Control Statement for NOCV Key-encrypting Keys

This example shows a control statement to load a key into the CKDS from a CEX2C or CEX3C, where the key is a key-encrypting key to be used as a NOCV KEK. The serial number of the card is 94000064. A key has been loaded on the card with the label ERC033.NOCV.IMPORTER.

```
OPKYLOAD LABEL(ERC033.NOCV.IMPORTER) SERNBR(94000064) NOCV
```

KGUP checks the CKDS for the label and will fail if the label exists. KGUP then queries the coprocessor to see if the key exists on the card. If the key exists, the key token is retrieved from the card. If the key is a key-encrypting key with the default control vector, the NOCV token flag is set. The token is then loaded into the CKDS.

Example 13 – ADD and UPDATE Control Statements with CLRDES and CLRAES Key Type

This example shows a control statement that adds a CLRDES key to the CKDS with a random 16 byte key. The ALGORITHM keyword is not allowed.

```
ADD TYPE(CLRDES) LENGTH(16) LAB(CLRDES.KEYLN8)
```

This example shows a control statement for updates a CLRAES key in the CKDS with a random 24 byte key. The ALGORITHM keyword is not allowed.

```
UPDATE TYPE(CLRAES) LENGTH(24) LAB(CLRAES.NOV11)
```

Example 14 – ADD and UPDATE Control Statement for a Group of CLRDES or CLRAES Keys with a Key Value

This example shows a control statement that adds a group of CLRDES. The clear key value is specified. The CLEAR and ALGORITHM keywords are not allowed.

```
ADD TYPE(CLRDES) KEY(2C2C2C2C2C2C2C2C,1616161616161616),
LAB(X.CLRDES.KEYLN16,Y.CLRDES.KEYLN16,Z.CLRDES.KEYLN16)
```

This example shows a control statement that updates a group of CLRAES. The clear key value is specified. The CLEAR and ALGORITHM keywords are not allowed.

```
UPDATE TYPE(CLRAES) KEY(2C2C2C2C2C2C2C2C,1616161616161616),
LAB(X.CLRAES.NOV11,Y.CLRAES.NOV11,Z.CLRAES.NOV11)
```

Example 15 – ADD and UPDATE Control Statements with ALGORITHM Keyword

This example shows a control statement that adds an AES DATA key to the CKDS with a random 128-bit key value.

```
ADD TYPE(DATA) ALGORITHM(AES) LENGTH(16) LAB(AES.BIT128)
```

This example shows a control statement that adds a group of AES DATA keys to the CKDS. A different key value will be generated for each label.

```
ADD TYPE(DATA) LENGTH(16) LAB(A.AES.L128,B.AES.L128,C.AES.L128) ALGORITHM(AES)
```

This example shows a control statement that adds a group of DES DATA keys to the CKDS. A different key value will be generated for each label.

```
ADD TYPE(DATA) LENGTH(16) LAB(A.DES.L16,B.DES.L16,C.DES.L16) ALGORITHM(DES)
```

This example shows a control statement that adds a group of DES DATA keys to the CKDS. A different key value will be generated for each label.

```
ADD TYPE(DATA) ALGORITHM(DES) LENGTH(24) RAN(DES.LN24.KEY1,DES.LN24.KEY3)
```

This example shows a control statement that changes an AES DATA key.

```
UPDATE TYPE(DATA) KEY(4343434343434343,5656565656565656),
LAB(AES.BIT128) ALGORITHM(AES)
```

This example shows a control statement that changes a range of DES keys.

```
UPDATE TYPE(DATA) LENGTH(16) RAN(DES.KEY1,DES.KEY3) ALGORITHM(DES)
```

Example 16 – ADD control statement to add a range of CLRDES keys

This example shows a control statement that adds a range of CLRDES keys. A different key value is generated for each key label.

```
ADD TYPE(CLRDES) LENGTH(24) RAN(CLRDES.KEYLN24.KEY1,CLRDES.KEYLN24.KEY3)
```

Example 17 – UPDATE control statement with CLRDES keyword

This example shows a control statement that changes a CLRDES key.

```
UPDATE TYPE(CLRDES) KEY(4343434343434343) LAB(CLRDES.KEYLN8)
```

Example 18 – UPDATE control statement with CLRDES keyword

This example shows a control statement that changes a range of CLRDES keys.

```
UPDATE TYPE(CLRDES) LENGTH(16) RAN(CLRDES.KEY1,CLRDES.KEY3)
```

Example 19 – DELETE control statement with CLRDES keyword

This example shows a control statement that deletes a CLRDES key.

```
DELETE TYPE(CLRDES) LAB(CLRDES.KEYLN24)
```

Example 20 – DELETE control statement to delete a group of CLRDES key labels

This example shows a control statement that deletes a group of CLRDES keys.

```
DELETE TYPE(CLRDES) LAB(A.KEYLN16,B.KEYLN16,C.KEYLN16)
```

Example 21 – RENAME Control Statement with CLRDES Keyword

This example shows a control statement that renames a CLRDES key.

```
RENAME TYPE(CLRDES) LAB(CLRDES.KEYLN16,CLRDES.DOUBLE.LENGTH.KEY)
```

Example 22 – ADD Control Statement with CLRAES Keyword

This example shows a control statement that adds a CLRAES key to the CKDS with a random 16 byte key.

```
ADD TYPE(CLRDES) LENGTH(16) LAB(AES.BIT128)
```

Example 23 – ADD Control Statement to Add a Group of CLRAES Keys

This example shows a control statement that adds a group of CLRAES keys to the CKDS. Key value is generated.

```
ADD TYPE(CLRAES) LENGTH(16) LAB(A.AES.L128,B.AES.L128,C.AES.L128)
```

Example 24 – ADD Control Statement to Add a Group of CLRAES Keys

This example shows a control statement that adds a group of CLRAES keys. The clear key value is specified.

```
ADD TYPE(CLRAES) KEY(2C2C2C2C2C2C2C2C,1616161616161616,A9A9A9A9A9A9A9A9),
LAB(X.AES.BIT192,Y.AES.BIT192,Z.AES.BIT192)
```

Example 25 – ADD Control Statement to Add a Range of CLRAES Keys

This example shows a control statement that adds a range of CLRAES keys. A different key value is generated for each key label.

```
ADD TYPE(CLRAES) LENGTH(32) RAN(AES.LN32.KEY1,AES.LN32.KEY3)
```

Example 26 – UPDATE Control Statement with CLRAES Keyword

This example shows a control statement that changes a CLRAES key.

```
UPDATE TYPE(CLRAES) KEY(4343434343434343,1616161616161616) LAB(AES.BIT128)
```

Example 27 – UPDATE Control Statement with CLRAES Keyword

This example shows a control statement that changes a range of CLRAES keys.

```
UPDATE TYPE(CLRAES) LENGTH(16) RAN(AES.KEY1,AES.KEY3)
```

Example 28 – DELETE Control Statement with CLRAES Keyword

This example shows a control statement that deletes a CLRAES key.

```
DELETE TYPE(CLRAES) LAB(AES.LN24)
```

Example 29 – DELETE Control Statement to Delete a Group of CLRAES Key Labels

This example shows a control statement that deletes a group of CLRAES keys.

```
DELETE TYPE(CLRAES) LAB(A.AES.LN16,B.AES.LN16,C.AES.LN16)
```

Example 30 – RENAME Control Statement with CLRAES Keyword

This example shows a control statement that renames a CLRAES key.

```
RENAME TYPE(CLRAES) LAB(AES.ESC001,AES.EXC001)
```

Example 31 – ADD Control Statement for ALGORITHM keyword

This example shows a control statement that adds an AES DATA key to the CKDS with a random 128-bit key value.

```
ADD TYPE(DATA) ALGORITHM(AES) LENGTH(16) LAB(AES.BIT128)
```

This example shows a control statement that adds a DES DATA key to the CKDS with a random 16-byte key value.

```
ADD TYPE(DATA) ALGORITHM(DES) LENGTH(16) LAB(DES.KEYLN16)
```

This example shows a control statement that adds a group of AES DATA keys to the CKDS. A different key value will generated for each label.

```
 ADD TYPE(DATA) LENGTH(16) LAB(A.AES.L128,B.AES.L128,C.AES.L128) ALGORITHM(AES)
```

This example shows a control statement that adds a group of DES DATA keys to the CKDS. A different key value will generated for each label.

```
ADD TYPE(DATA) LENGTH(16) LAB(A.DES.L16,B.DES.L16,C.DES.L16) ALGORITHM(DES),
CLEAR
```

This example shows a control statement that adds a group of AES DATA keys. The clear key value is specified.

```
ADD TYPE(DATA) ALGORITHM(AES) CLEAR,
KEY(2C2C2C2C2C2C2C2C,1616161616161616,A9A9A9A9A9A9A9A9),
LAB(X.AES.BIT192,Y.AES.BIT192,Z.AES.BIT192)
```

This example shows a control statement that adds a group of DES DATA keys to the CKDS. A different key value will generated for each label.

```
ADD TYPE(DATA) ALGORITHM(DES) LENGTH(24) RAN(DES.LN24.KEY1,DES.LN24.KEY3)
```

Example 32 – UPDATE Control Statement with the ALGORITHM keyword

This example shows a control statement that changes an AES DATA key.

```
UPDATE TYPE(DATA) KEY(4343434343434343,5656565656565656),
LAB(AES.BIT128) ALGORITHM(AES)
```

This example shows a control statement that changes a range of DES keys.

```
UPDATE TYPE(DATA) LENGTH(16) RAN(DES.KEY1,DES.KEY3) ALGORITHM(DES)
```

Specifying KGUP data sets

During key generator utility program (KGUP) processing, you store the information you supply and receive in these data sets:

- The cryptographic key data set (CKDS) contains key entries that you have KGUP add, update, rename, or delete.
- The control statement input data set contains the control statements that specify the functions you want KGUP to perform.
- The diagnostics data set contains information you can use to check that the control statement succeeded.
- The key output data set contains information that another system uses to create keys that are complements of keys on your system.
- The control statement data set contains control statements that another system uses to create keys that are complements of keys on your system.

You specify the names of the data sets in the job control language to submit the job.

These topics describe the data sets that KGUP accesses or generates in detail.

Cryptographic Key Data Set (CKDS)

This VSAM key sequenced data set contains the cryptographic keys for a particular KGUP job.
There are two types of CKDS:

- Fixed-length record format. The logical record length (LRECL) is 252 bytes.
- Variable-length record format. The logical record length (LRECL) is 1024 bytes.

The first record in the CKDS is a header record. The header record is the same for both types of CKDS.

Note:

The records in the fixed-length record format CKDS are in this format:

Key label
(Character length 64 bytes) The key label specified on the control statement.

Key type
(Character length 8 bytes) The key type specified on the control statement.

Creation date
(Character length 8 bytes) The initial date the record was created, in the format `YYYYMMDD`.

Creation time
: (Character length 8 bytes) The initial time the record was created, in the format HHMMSSTH.

Last update date
: (Character length 8 bytes) The most recent date the record was updated, in the format YYYYMMDD.

Last update time
: (Character length 8 bytes) The most recent time the record was updated, in the format HHMMSSTH.

Key token
: (Character length 64 bytes) A key token is composed of the key value and control information. The master key encrypts the key value in this field. For a description of format of a key token, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

CKDS flag bytes
: (Bit length 2 bytes) If bit zero is set to one, the key within the token is a partial key. All the other bits are reserved.

Reserved
: (Character length 26 bytes) Reserved. This field contains binary zeros.

Installation Data
: (Character length 52 bytes) Using the KGUP exit, conversion program exit, or single-record, Single-record, read-write exit, you can place information associated with the key entry into this field.

Authentication code
: (Character length 4 bytes) The message authentication code computed on the previous fields of the record using a system key that is a MAC generation key. ICSF uses the code to verify the record when the record is updated.

The records in the variable-length record format CKDS are in this format:

Key label
: (Character length 64 bytes) The key label specified on the control statement.

Key type
: (Character length 8 bytes) The key type specified on the control statement.

Creation date
: (Character length 8 bytes) The initial date the record was created, in the format YYYYMMDD.

Creation time
: (Character length 8 bytes) The initial time the record was created, in the format HHMMSSTH.

Last update date
: (Character length 8 bytes) The most recent date the record was updated, in the format YYYYMMDD.

Last update time
: (Character length 8 bytes) The most recent time the record was updated, in the format HHMMSSTH.

Record length
(Integer length 4 bytes) The length of the record

Reserved
(Character length 60 bytes) Reserved. This field contains binary zeros.

CKDS flag bytes
(Bit length 2 bytes) If bit zero is set to one, the key within the token is a partial key. If bit three is set to one, the key token in the record is a variable-length token. All the other bits are reserved.

Reserved
(Character length 26 bytes) Reserved. This field contains binary zeros.

Installation Data
(Character length 52 bytes) Using the KGUP exit, conversion program exit, or single-record, single-record, read-write exit, you can place information associated with the key entry into this field.

Authentication code
(Character length 20 bytes) The message authentication code computed on the record with the authentication code field set to binary zeros. ICSF uses the code to verify the record when the record is updated.

Key token
(Character length variable) A key token is composed of the key value and control information. The master key encrypts the key value in this field. For a description of format of a key token, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

The header record in the CKDS is in this format:

Key label
(Character length 64 bytes) Binary zeros. This field is not to be used.

Key type
(Character length 8 bytes) Binary zeros. This field is not to be used.

Creation date
(Character length 8 bytes) The initial date the record was created, in the format `YYYYMMDD`.

Creation time
(Character length 8 bytes) The initial time the record was created, in the format `HHMMSSTH`.

Last update date
(Character length 8 bytes) The most recent date the record was updated, in the format `YYYYMMDD`.

Last update time
(Character length 8 bytes) The most recent time the record was updated, in the format `HHMMSSTH`.

Sequence number
(Character length 2 bytes) Initially binary zero, increments each time the data set is processed.

CKDS header flag bytes
(Bit length 2 bytes) If bit zero is set to one, the DES master key verification pattern is valid. If bit one is set to one, the DES master key authentication pattern is valid. If bit two is set to one, the AES master key verification pattern is valid. If bit 8 is set to one, record

authentication has been disabled. If bit 9 is set to one, the format of the CKDS is variable-length record format. All the other bits are reserved.

DES master key verification pattern
(Character length 8 bytes) The DES master key verification pattern. When you initialize the CKDS and master key or change the master key, ICSF calculates a verification pattern and places it into this field. ICSF calculates the verification pattern by using the current master key and the verification algorithm that is described in

DES master key authentication pattern
(Character length 8 bytes) The DES master key authentication pattern. Only used on system with the Cryptographic Coprocessor Feature.

When you initialize the CKDS and master key or change the master key, ICSF calculates an authentication pattern and places it into this field. ICSF calculates the authentication pattern by using the current master key and the authentication pattern algorithm.

AES master key verification pattern
(Character length 8 bytes) The AES master key verification pattern.

When you initialize the CKDS and AES master key or change the AES master key, ICSF calculates a verification pattern and places it into this field. ICSF calculates the verification pattern by using the current master key and the verification algorithm.

Reserved
(Character length 64 bytes) Reserved. This field contains binary zeros.

Installation Data
(Character length 52 bytes) Using the KGUP installation exit, you can place information associated with the key entry into this field.

Authentication code
(Character length 4 bytes) The message authentication code computed on the previous fields of the record using a system key that is a MAC generation key. ICSF creates the code when ICSF creates the system keys at CKDS initialization. ICSF uses the code to verify the CKDS when the CKDS is read.

The variable-length record format of the CKDS does not generate the authentication code for the header record.

In the KGUP job stream, it is defined by the CSFCKDS data definition statement.

Control Statement Input Data Set

This data set contains the control statements that the particular KGUP job processes. For a description of the syntax of these control statements, see "Using KGUP control statements" on page 189.

This data set is a physical sequential data set with a fixed logical record length (LRECL) of 80 bytes.

Note: If a control statement adds or updates a key, later control statements in the control statement input data set for that KGUP job use the new or updated key.

In the KGUP job stream, the control statement input data set is defined by the CSFIN data definition statement.

Diagnostics Data Set

This data set contains a copy of each input control statement that is followed by one or more diagnostic messages that were generated for that control statement. It is a physical sequential data set with a fixed logical record length (LRECL) of 133 bytes. It should be fixed with ASA codes. Figure 56 shows an example of a diagnostics data set.

```
KEY GENERATION DIAGNOSTIC REPORT  DATE:1997/9/14 (YYYY/MM/DD) TIME:12:10:15 PAGE 1
 /* THIS IS A KEY USED TO EXPORT KEYS FROM A TO B */
 ADD TYPE(EXPORTER) TRANSKEY(TK1),
  LABEL(ATOB)
 > > > CSFG0321 STATEMENT SUCCESSFULLY PROCESSED.
 /* THIS IS A KEY USED TO IMPORT KEYS FROM B TO A */
 ADD TYPE(IMPORTER) TRANSKEY(TK1),
  LABEL(BTOA)
 > > > CSFG0321 STATEMENT SUCCESSFULLY PROCESSED.
 > > > CSFG0780 A REFRESH OF THE IN-STORAGE CKDS IS NECESSARY TO ACTIVATE CHANGES MADE BY KGUP.
 > > > CSFG0002 CRYPTOGRAPHIC KEY GENERATION - END OF JOB. RETURN CODE = 0.
```

Figure 56. Diagnostics Data Set Example

In the KGUP job stream, the data set is defined by the CSFDIAG data definition statement.

Key Output Data Set

This data set contains information about each key KGUP generates, except an importer key used to protect a key that is stored with a file. Each entry contains the key value and the complement key type of the key created. Another system can use this information to create a key that is the complement of the key your system created.

This data set is a physical sequential data set with a fixed logical record length (LRECL). The minimum LRECL is 208 bytes. This will accommodate 64 byte DES key tokens. If you are exporting AES keys that use the variable-length key token, the LRECL should be at least 500. The maximum supported LRECL is 1044.

To establish key exchange with a system that does not use KGUP control statements, you can send that system information from this data set. The receiving system can then use this information to create the complement of the key you created. You can print or process this data set when KGUP ends.

KGUP only lists a record for the key if the TRANSKEY or CLEAR keyword was in the control statement. If the TRANSKEY keyword was specified in the output key data set, KGUP lists, for the key type, the complement of the control statement key type. KGUP lists, for the key value, the key encrypted under the transport key as specified by the TRANSKEY keyword.

The encrypted key is in the form of an external key token. An external key token contains the encrypted key value and control information about the key. For example, the token contains the control vector for the key type or the associated data.

If the CLEAR keyword was specified, in the output key data set KGUP lists, for the key type, the complement of the control statement key type. KGUP lists, for the key value, the clear key value of the key. With this information another system could generate keys that are complements of the keys your system generated. This would permit your system and the other system to exchange keys.

When KGUP generates two complementary keys, each encrypted by a different transport key, KGUP lists a record for each key. The first record contains a key that is encrypted under the first transport key variant and the type that is specified on the control statement. The second record contains a key that is encrypted under the second transport key variant and a type that is the complement of the first key.

The records in the key output data set are in this format:

Key label
: (Character length 64 bytes) The key label specified on the control statement.

Key type
: (Character length 8 bytes) The key type specified on the control statement or the complement of that key type if the TRANSKEY keyword was specified.

TRANSKEY label or `CLEAR`
: (Character length 64 bytes) Either the key label of a transport key which encrypts the key entry or the character string `CLEAR` (left justified) if the key is unencrypted.

TRANSKEY type
: (Character length 8 bytes) The key type of the TRANSKEY, which is always exporter.

Key Token
: (Character length variable bytes) A key token is composed of the key value and control information. The key value in this field is either unencrypted or encrypted under a transport key. For DES keys, the clear key value is stored where the key value is stored in a fixed length token. For AES keys, the clear key value is stored, left justified, in the key token field. For version '00'x and '01'x DES key tokens, the token is always 64 bytes long. For version '05'x AES key tokens, the token is variable length. The length field is a two byte field at offset 2 in the token. For a description of format of a key token, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

In the KGUP job stream, the data set is defined by the CSFKEYS data definition statement.

Control Statement Output Data Set

KGUP produces an output control statement for every key that is generated as a result of an input control statement with the TRANSKEY keyword specified. The output control statement contains the complement key type of the key type that is specified on the input control statement. The value that is output for the KEY keyword is encrypted under the transport key that is specified on the input control statement.

You can edit the output control statements and distribute them to the appropriate sites for input to KGUP at those locations.

The data set is a physical sequential data set with a fixed logical record length (LRECL) of 80 bytes.

One output control statement appears when you have KGUP generate a key value and create an operational and exportable key pair using a transport key.

Two output control statements appear when you have KGUP generate two exportable keys by using two different transport keys. These statements generate complementary keys types. You can send each statement to a different site to establish communication between the two sites.

In the KGUP job stream, the data set is defined by the CSFSTMNT data definition statement. The data set will contain information only when the input control statement contains the TRANSKEY keyword. The TRANSKEY keyword indicates that you will be transporting the key to another system.

The specific name of these types of data sets must appear in the job stream that runs KGUP.

Submitting a job stream for KGUP

The key generator utility program (KGUP) is an APF-authorized program that runs as a batch job. It requires certain JCL statements to run. Submit the JCL to run KGUP when you create the KGUP control statements and data sets.

The JCL to run KGUP should be in this format:

```
//KGUPPROC EXEC PGM=CSFKGUP,PARM=('SSM')
//CSFCKDS  DD   DSN=PROD.CKDS,DISP=OLD
//CSFIN    DD   DSN=PROD.KGUPIN.GLOBAL,DISP=OLD
//CSFDIAG  DD   DSN=PROD.DIAG.GLOBAL,DISP=OLD
//CSFKEYS  DD   DSN=PROD.KEYS.GLOBAL,DISP=OLD
//CSFSTMNT DD   DSN=PROD.STMT.GLOBAL,DISP=OLD
//
```

Figure 57. KGUP Job Stream

The EXEC statement specifies the load module name for KGUP. The PARM keyword on the EXEC statement passes information to KGUP. The keyword specifies either:

- NOSSM to indicate that special secure mode must be disabled
- SSM to indicate that special secure mode must be enabled

You must pass the SSM parameter if any KGUP control statements for the KGUP run contain the CLEAR keyword. NOSSM is the default.

If special secure mode is not enabled and you pass the SSM parameter to KGUP, the program ends immediately without processing any KGUP control statements. If you pass the NOSSM parameter and KGUP encounters a control statement with the CLEAR keyword, the job ends immediately.

In the JCL example, the PARM keyword specifies SSM to indicate that special secure mode should be enabled. You specify SSM if any control statement in the control statement input data set, PROD.KGUPIN.GLOBAL, contains the CLEAR keyword.

In the JCL, the data definition (DD) statements name the data sets necessary to input information to KGUP and output information from the program. See "Specifying KGUP data sets" on page 224 for a detailed description of these data sets.

Attention: If a KGUP job ends prematurely, results of the job are unpredictable. You should not read that cryptographic key data set into storage for use.

For a description of the KGUP return codes, see the explanation of message CSFG0002, which is in *z/OS Cryptographic Services ICSF Messages*.

Enabling Special Secure Mode

When you pass the SSM parameter to KGUP in a JCL statement, you need to enable special secure mode processing. To use special secure mode, the installation options data set must specify YES for the SSM installation option or the CSF.SSM.ENABLE SAF profile must be defined in the XFACILIT SAF resource class.

Running KGUP Using the MVS/ESA Batch Local Shared Resource (LSR) Facility

The MVS/ESA batch LSR subsystem improves performance for random access file processing by reducing the number of inputs and outputs to VSAM data sets. Batch LSR allows a program to use local shared resources rather than non-shared resources. For information about the batch LSR subsystem, see the z/OS MVS, MVS Batch Local Shared Resources information in the appropriate z/OS Information Center, which may be selected from http://www-03.ibm.com/systems/z/os/zos/bkserv/.

VSAM provides a deferred write option on VSAM ACB processing when a program uses shared resources.

By using the batch LSR subsystem and the VSAM deferred write option together, you may improve KGUP performance when adding many keys, for example 10,000 keys, to the CKDS. If your installation has batch LSR and VSAM deferred write, you may improve performance when adding a large number of keys by using different JCL in the KGUP job stream.

Instead of using this CSFCKDS DD statement:

```
//CSFCKDS DD DSN=cryptographic-key-data-set-name,DISP=OLD
```

Use these statements:

```
//CSFALT  DD DSN=cryptographic-key-data-set-name,DISP=OLD
//CSFCKDS DD SUBSYS=(BLSR,'DDNAME=CSFALT',
//           'DEFERW=YES')
```

You should specify a large amount of storage for the REGION parameter (for example, REGION=32M) on the JOB or EXEC JCL statement. The rest of the JCL statements to run the KGUP job should be in the format that is shown in Figure 57 on page 230.

Reducing Control Area Splits and Control Interval Splits from a KGUP Run

KGUP processes keys on a disk copy of a CKDS which is a VSAM data set. KGUP uses key-direct update processing to process the keys. To access keys, VSAM uses the key's label as the VSAM key. This means that keys are added 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, E and you add C (C must be placed between B and E). These splits can leave considerable free space in the data set.

The amount of control area splits and control interval splits in the CKDS affects performance. You may want to periodically use the TSO LISTCAT command to list information about the number of control area splits and control interval splits in a CKDS.

You can help reduce the frequency of control interval and control area splits by ensuring that key generator utility control statements are always in the correct collating sequence, A-Z, 0-9, if possible. When adding keys to a new CKDS, add the key entries in sequential order. Also, when adding new entries to the CKDS, you can reorganize the data set to reduce control area splits and control interval splits. To do this, copy the disk copy of the CKDS into another disk copy using the AMS REPRO command or AMS EXPORT/IMPORT commands. You may want to reorganize the data set after every KGUP run.

Note: If it is practical, you may want to perform this procedure to reduce control area splits. If you are inserting a large number of keys in the middle of a CKDS, you may want to remove and save all the keys after the place in the data set where you are inserting the keys. In this way, you are adding the keys to the end rather than the middle of the data set. When you finish adding the keys, place the keys that you removed back in the data set.

For a detailed explanation of keyed-direct update processing and a description of what happens when control area and control interval splits occur, refer to *z/OS DFSMS Access Method Services Commands*.

Refreshing the In-Storage CKDS

ICSF functions access an in-storage copy of the CKDS when the functions reference keys by label. However when you use KGUP, the program makes changes to a disk copy of the CKDS. This situation allows you to maintain the keys in the data set without disturbing current cryptographic operations.

When you update the disk copy, you can use the Refresh option on the Key Administration panel to replace the in-storage copy with the disk copy. For a description of this panel path, see “Steps for refreshing the active CKDS using the ICSF panels” on page 257. Besides using the panels to refresh the in-storage CKDS, you can invoke a utility program to perform the task. Refer to “Refreshing the in-storage CKDS using a utility program” on page 345 for details.

The preferred method for performing a CKDS refresh is to use the coordinated refresh function. See “Performing a coordinated CKDS refresh” on page 136 for additional information.

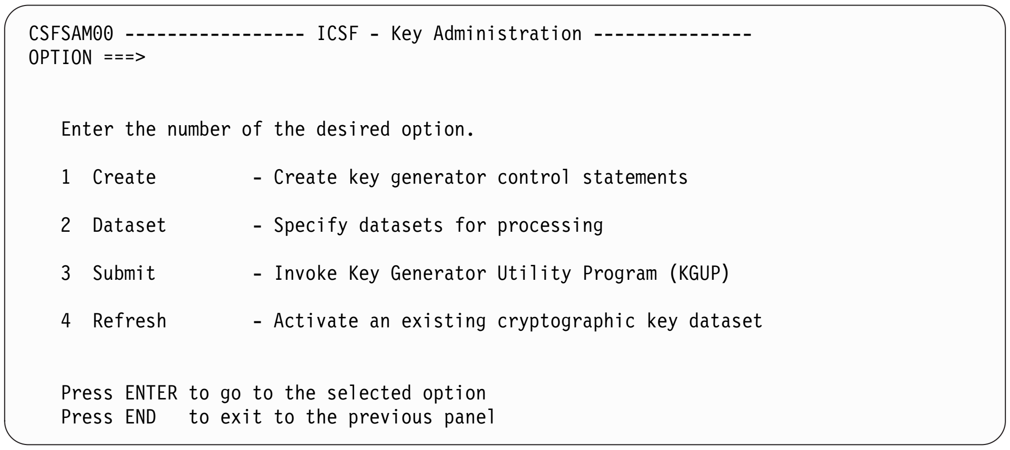
Using KGUP Panels

The key generator utility program (KGUP) panels help you run KGUP by providing panels to do these tasks:
- Create KGUP control statements (except OPKYLOAD).
- Specify the data sets for KGUP processing.
- Invoke KGUP by submitting job control language (JCL) statements.
- Replace the in-storage copy of the cryptographic key data set (CKDS) with the disk copy that KGUP processing changed.

Using the panels, you can perform the tasks to use KGUP to generate or receive keys for PIN and key distribution and to maintain the CKDS.

To access the KGUP panels, select option 8, KGUP, on the "ICSF Primary Menu panel" on page 387.

The Key Administration panel appears. See Figure 58.



```
CSFSAM00 ----------------- ICSF - Key Administration ---------------
OPTION ===>


   Enter the number of the desired option.

   1  Create        - Create key generator control statements

   2  Dataset       - Specify datasets for processing

   3  Submit        - Invoke Key Generator Utility Program (KGUP)

   4  Refresh       - Activate an existing cryptographic key dataset


   Press ENTER to go to the selected option
   Press END   to exit to the previous panel
```

Figure 58. Key Administration Panel

This panel allows you to access panels to perform the tasks to run KGUP. These topics describe the KGUP tasks.

Steps for creating KGUP control statements using the ICSF panels

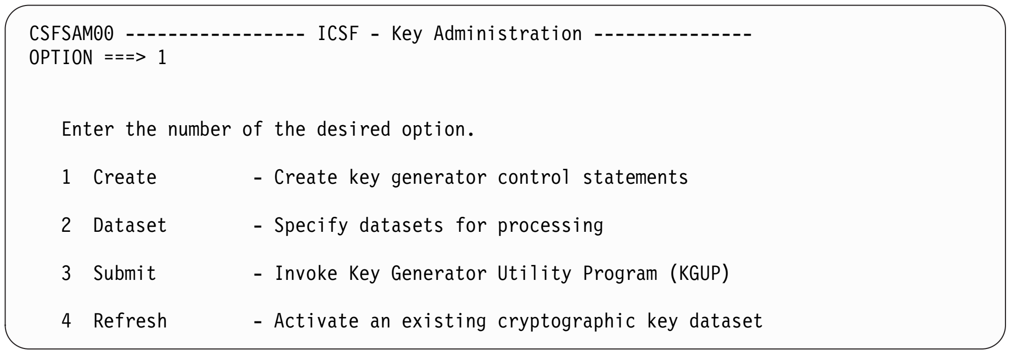
You create the control statements to specify the functions you want KGUP to perform. When you create the control statements, ICSF stores the statements in the control statement input data set.

When you create the control statements, do one of these procedures:

- Process the control statements by running KGUP.
- Do not process the control statements and just save the statements in the data set. Then at another time you can access the data set to add more control statements and submit the data set for KGUP processing.

To create the KGUP control statements:

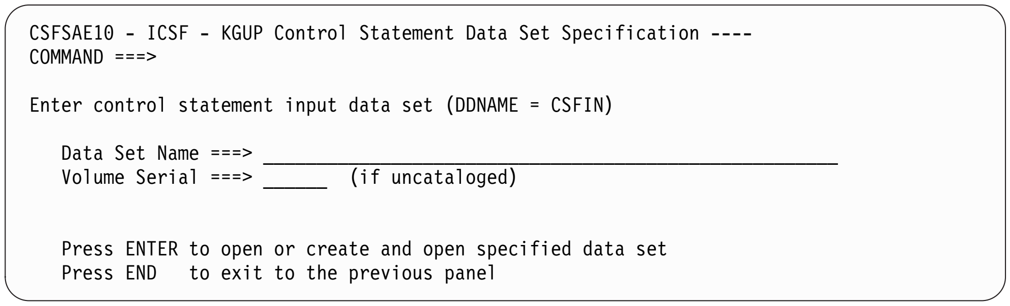
1. Select option 1, Create, on the Key Administration panel, as shown in Figure 59 on page 234, and press ENTER.



```
CSFSAM00 ----------------- ICSF - Key Administration ---------------
OPTION ===> 1


   Enter the number of the desired option.

   1  Create        - Create key generator control statements

   2  Dataset       - Specify datasets for processing

   3  Submit        - Invoke Key Generator Utility Program (KGUP)

   4  Refresh       - Activate an existing cryptographic key dataset
```

Figure 59. Selecting the Create Option on the Key Administration Panel

The KGUP Control Statement Data Set Specification panel appears. See Figure 60.



```
CSFSAE10 - ICSF - KGUP Control Statement Data Set Specification ----
COMMAND ===>

Enter control statement input data set (DDNAME = CSFIN)

   Data Set Name ===> ______________________________________________________
   Volume Serial ===> ______  (if uncataloged)


   Press ENTER to open or create and open specified data set
   Press END   to exit to the previous panel
```

Figure 60. KGUP Control Statement Data Set Specification Panel

2. Enter the name of the data set that you want to contain the control statements for KGUP processing.
   a. For partitioned data sets, specify a member name as part of the data set name.
   b. If the data set is not cataloged, you must also specify the volume serial for the data set in the Volume Serial field. This volume serial allows ICSF to access the correct volume when ICSF opens the data set.

   Note: If you specify NOPREFIX in your TSO profile, so data sets are not automatically prefixed with your userid, you must specify the fully qualified data set name within apostrophes. If you specify PREFIX without a valid prefix, your TSO userid becomes the prefix.

   Depending on your requirements, there are several options to choose from when entering the data set name. Refer to Table 50 for a list of these options and the steps to follow for each.

Table 50. Data Set Name Options

Option	**Steps**
To have KGUP append the control statements to an existing data set when you know the data set name and the member name	1. Specify the data set name and member name of the existing data set and press ENTER. The KGUP Control Statement Menu appears. See Figure 64 on page 237. The new control statements will be appended when any existing control statements in the data set.

Table 50. Data Set Name Options (continued)

Option	Steps
To have KGUP append the control statements to an existing data set when you know the data set name but not the member name	1. Specify the data set name of the existing data set and press ENTER. If the partitioned data set is not empty, the Member Selection List appears. See Figure 62 on page 236. 2. On the Member Selection List panel: • To select a member that already exists, place an `s` to the left of the member name in the list and press ENTER. For example, in Figure 62 on page 236 SHIFT2 is selected so the data set LARSON.CSFIN.TESTDS1P(SHIFT2) becomes the input control statement data set. • To locate a member on the selection list, type an `l` (the lowercase letter L) and the member name on the command line and press ENTER. The list moves so the member appears on the top line of the list and the cursor appears to the left of the member. • To create a new member, type `s` and the new member name on the command line and press ENTER. The KGUP Control Statement Menu appears. See Figure 64 on page 237. The new control statements will be appended when any existing control statements in the data set.
To have KGUP create a new data set	1. Specify a name for the new data set and press ENTER. The Allocation panel appears. See Figure 63 on page 236. 2. Enter the necessary information to allocate a new data set and press ENTER. The KGUP Control Statement Menu appears. See Figure 64 on page 237. The new control statements will be stored in the new data set.

Figure 61 shows an example of the KGUP Control Statement Data Set Specification panel with the partitioned data set CSFIN.TESTDS1P and a member name of TEST1.

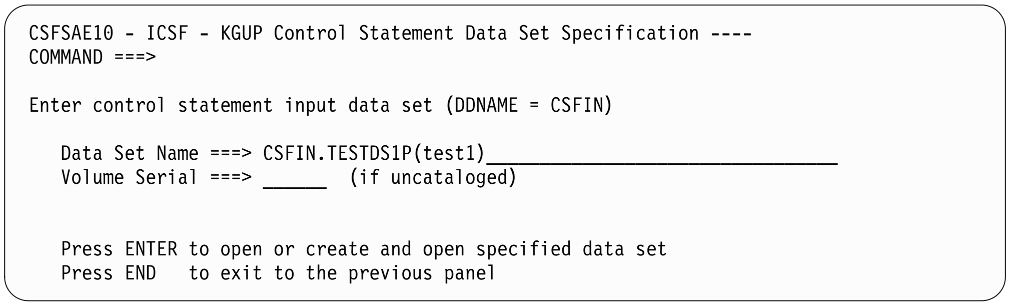
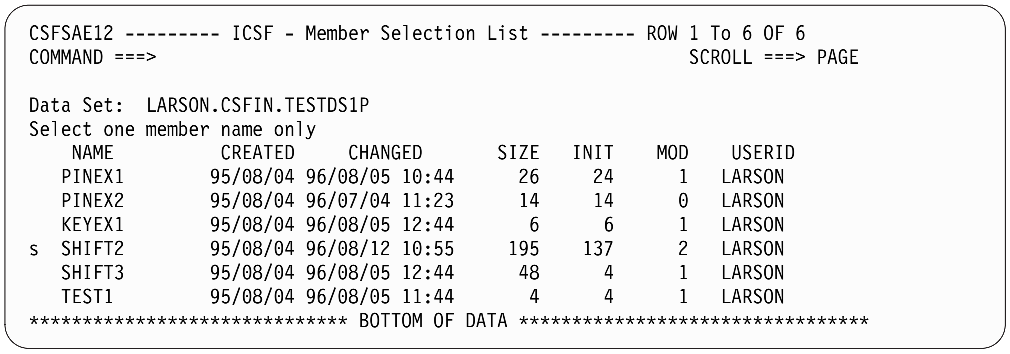

```
CSFSAE10 - ICSF - KGUP Control Statement Data Set Specification ----
COMMAND ===>

Enter control statement input data set (DDNAME = CSFIN)

   Data Set Name ===> CSFIN.TESTDS1P(test1)_________________________________
   Volume Serial ===> ______  (if uncataloged)


   Press ENTER to open or create and open specified data set
   Press END   to exit to the previous panel
```

Figure 61. Entering a Data Set Name on the KGUP Control Statement Data Set Specification Panel

If the member TEST1 did not previously exist, ICSF creates the member. If the member already exists, ICSF appends the control statements to the end of the data set. <Prefix>.CSFIN.TESTDS1P(test1) becomes the control statement input data set.

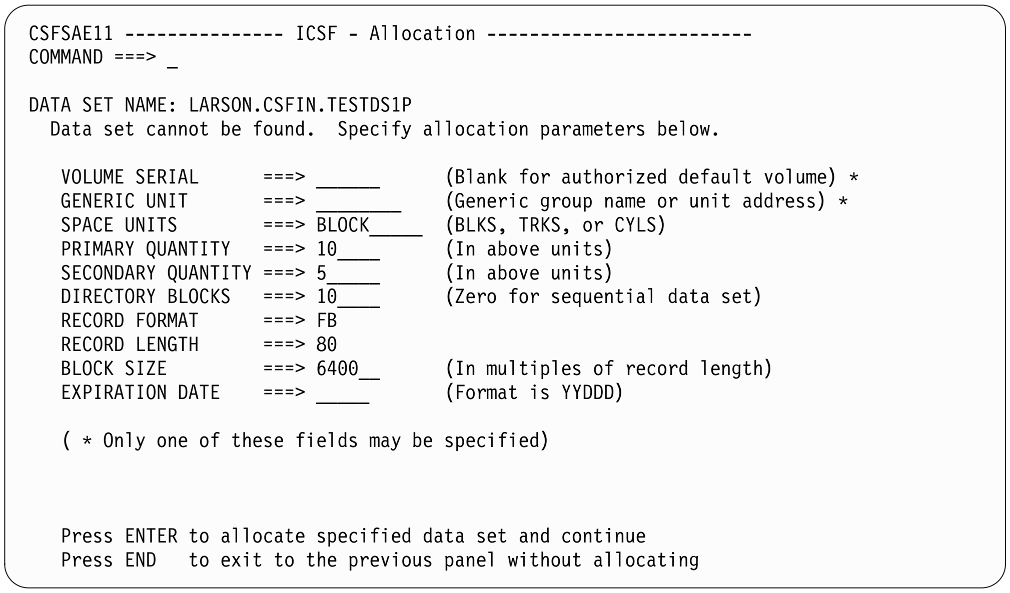
If you specify CSFIN.TESTDS1P without the member name, the Member Selection List panel appears. See Figure 62.



```
CSFSAE12 --------- ICSF - Member Selection List --------- ROW 1 To 6 OF 6
COMMAND ===>                                                  SCROLL ===> PAGE

Data Set:  LARSON.CSFIN.TESTDS1P
Select one member name only
    NAME          CREATED     CHANGED       SIZE   INIT    MOD    USERID
   PINEX1       95/08/04 96/08/05 10:44      26     24      1   LARSON
   PINEX2       95/08/04 96/07/04 11:23      14     14      0   LARSON
   KEYEX1       95/08/04 96/08/05 12:44       6      6      1   LARSON
s  SHIFT2       95/08/04 96/08/12 10:55     195    137      2   LARSON
   SHIFT3       95/08/04 96/08/05 12:44      48      4      1   LARSON
   TEST1        95/08/04 96/08/05 11:44       4      4      1   LARSON
****************************** BOTTOM OF DATA *********************************
```

Figure 62. Member Selection List Panel

If you specify a new data set name, the Allocation panel appears. See Figure 63.



```
CSFSAE11 --------------- ICSF - Allocation -------------------------
COMMAND ===> _

DATA SET NAME: LARSON.CSFIN.TESTDS1P
  Data set cannot be found.  Specify allocation parameters below.

   VOLUME SERIAL      ===> ______      (Blank for authorized default volume) *
   GENERIC UNIT       ===> ________    (Generic group name or unit address) *
   SPACE UNITS        ===> BLOCK_____  (BLKS, TRKS, or CYLS)
   PRIMARY QUANTITY   ===> 10____      (In above units)
   SECONDARY QUANTITY ===> 5_____      (In above units)
   DIRECTORY BLOCKS   ===> 10____      (Zero for sequential data set)
   RECORD FORMAT      ===> FB
   RECORD LENGTH      ===> 80
   BLOCK SIZE         ===> 6400__      (In multiples of record length)
   EXPIRATION DATE    ===> _____       (Format is YYDDD)

   ( * Only one of these fields may be specified)



   Press ENTER to allocate specified data set and continue
   Press END   to exit to the previous panel without allocating
```

Figure 63. Entering Data Set Information on the Allocation Panel

Once the data set has been selected or created, the data set becomes the control statement input data set on the KGUP Control Statement Menu, as shown in Figure 64 on page 237. The name of the control statement input data set you specified appears at the top of the panel.

From this panel, you can press END to go back to the KGUP Control Statement Data Set Specification panel. On the later panel you can either specify another data set to store control statements, or press END again to return to the Key Administration panel.

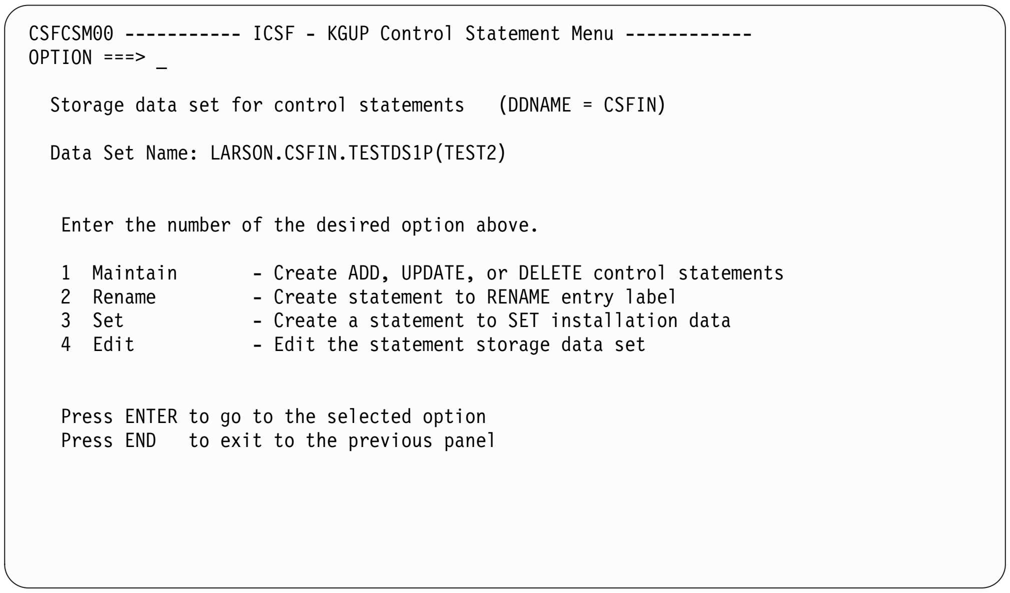

```
CSFCSM00 ----------- ICSF - KGUP Control Statement Menu ------------
OPTION ===> _

  Storage data set for control statements   (DDNAME = CSFIN)

  Data Set Name: LARSON.CSFIN.TESTDS1P(TEST2)


   Enter the number of the desired option above.

   1  Maintain       - Create ADD, UPDATE, or DELETE control statements
   2  Rename         - Create statement to RENAME entry label
   3  Set            - Create a statement to SET installation data
   4  Edit           - Edit the statement storage data set


   Press ENTER to go to the selected option
   Press END   to exit to the previous panel
```

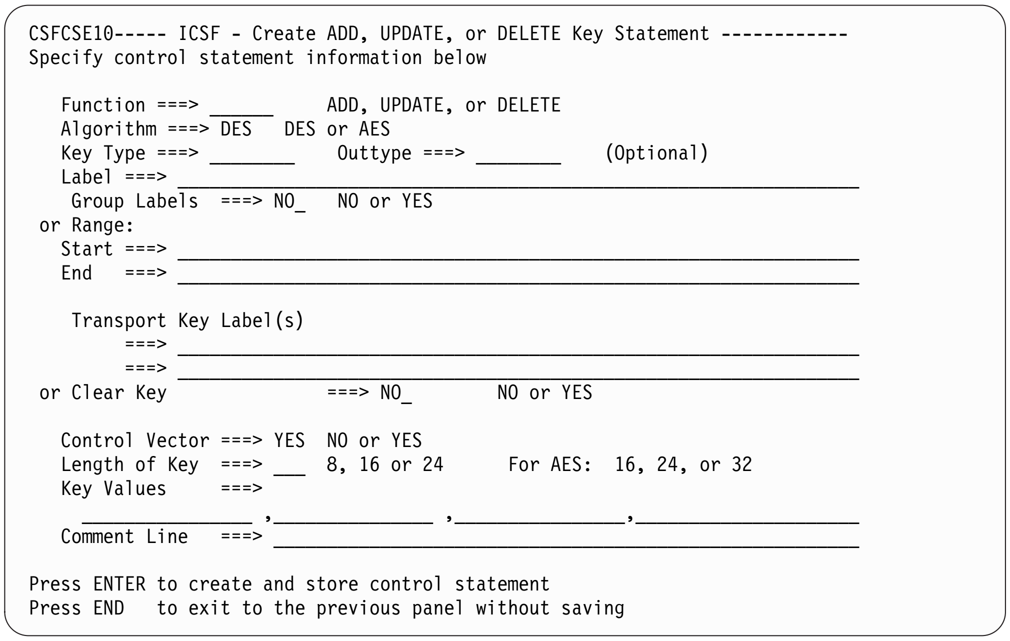
Figure 64. KGUP Control Statement Menu Panel

3. Choose the type of control statement you want to create and press ENTER.
   - To create an ADD, UPDATE, or DELETE control statement, select option 1. For information, see “Steps for creating ADD, UPDATE, or DELETE control statements.”
   - To create a RENAME control statement, select option 2. For information, see “Steps for creating a RENAME control statement” on page 244.
   - To create a SET control statement, select option 3. For information, see “Steps for creating a SET control statement” on page 247.
   - To edit the input control statement data set, select option 4. For information, see “Steps for editing control statements” on page 249.

   When you choose the Maintain, Rename, or Set option, you access the panels to create the control statement you want. When you create a control statement, the statement is placed in the specified control statement input data set. To edit the control statements that are stored in this data set, choose the Edit option.

Steps for creating ADD, UPDATE, or DELETE control statements

When you select Maintain (option 1) on the KGUP Control Statement Menu panel, the Create ADD, UPDATE, or DELETE Key Statement panel appears. See Figure 65 on page 238.



```
CSFCSE10----- ICSF - Create ADD, UPDATE, or DELETE Key Statement ------------
Specify control statement information below

   Function ===> ______     ADD, UPDATE, or DELETE
   Algorithm ===> DES   DES or AES
   Key Type ===> ________     Outtype ===> ________    (Optional)
   Label ===> ________________________________________________________________
    Group Labels  ===> NO_   NO or YES
 or Range:
   Start ===> ________________________________________________________________
   End   ===> ________________________________________________________________

    Transport Key Label(s)
        ===> ________________________________________________________________
        ===> ________________________________________________________________
 or Clear Key              ===> NO_        NO or YES

   Control Vector ===> YES  NO or YES
   Length of Key  ===> ___  8, 16 or 24      For AES:  16, 24, or 32
   Key Values     ===>
     _______________ ,_______________ ,_______________,__________________
   Comment Line   ===> _______________________________________________________

Press ENTER to create and store control statement
Press END   to exit to the previous panel without saving
```

Figure 65. Create ADD, UPDATE, or DELETE Key Statement Panel

1. On the panel, fill out the fields to create the ADD, UPDATE, or DELETE control statement that you want KGUP to process. Each field on the panel corresponds to a control statement keyword. The panel helps you to create a complete, syntactically correct ADD, UPDATE, or DELETE control statement.

   The panel creates control statements according to the syntax described in "Syntax of the ADD and UPDATE control statements" on page 191. See that topic for more information about the control statement keywords.
2. In the Function field, select the function you want KGUP to perform.

   Function
   Result

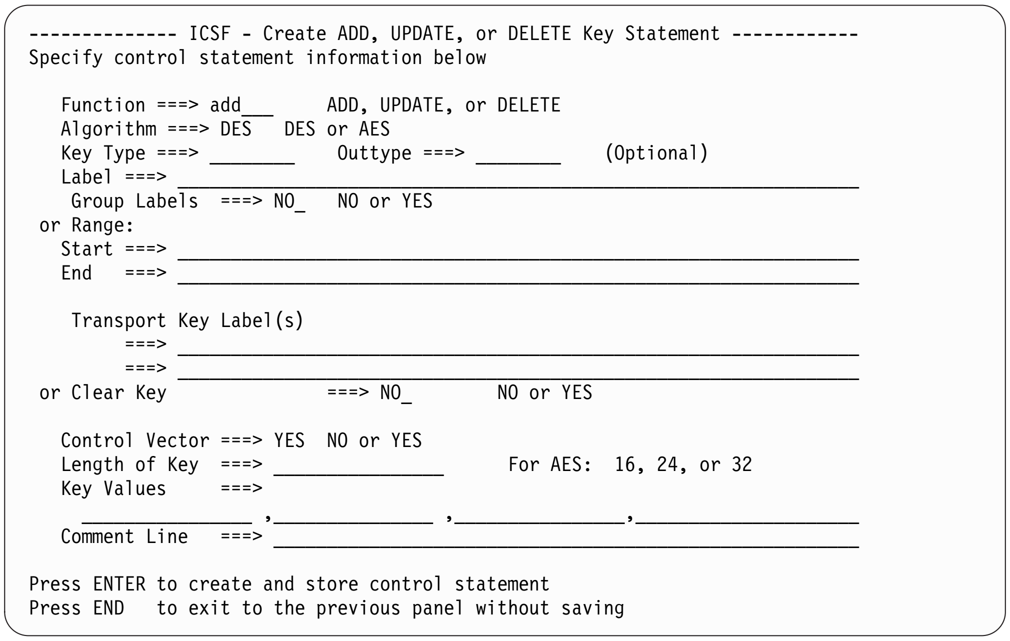
   ADD Enter new key entries in the CKDS. Generate and receive key values for key distribution.

   UPDATE
   Change existing entries in the CKDS. Generate and receive key values for key distribution.

   DELETE
   Remove entries from the CKDS.

   You can just type the first letter of the function in the first position in a field on the panel. For example, in Figure 66 on page 239, a was entered in the Function field to specify the ADD function. ICSF recognizes the abbreviation.

   For a description of the keywords you must specify for each function, see "Using the ADD and UPDATE control statements for key management and distribution functions" on page 208.

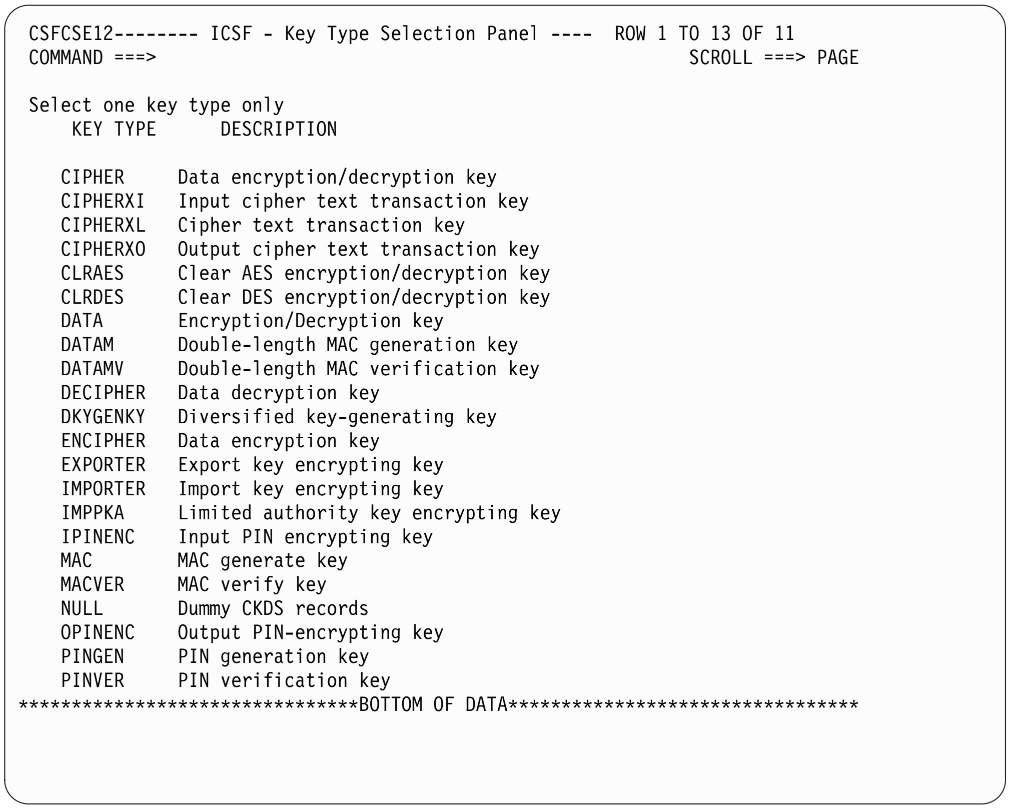


```
-------------- ICSF - Create ADD, UPDATE, or DELETE Key Statement ------------
Specify control statement information below

   Function ===> add___     ADD, UPDATE, or DELETE
   Algorithm ===> DES   DES or AES
   Key Type ===> ________     Outtype ===> ________    (Optional)
   Label ===> ________________________________________________________________
    Group Labels  ===> NO_   NO or YES
 or Range:
   Start ===> ________________________________________________________________
   End   ===> ________________________________________________________________

    Transport Key Label(s)
         ===> ________________________________________________________________
         ===> ________________________________________________________________
 or Clear Key              ===> NO_        NO or YES

   Control Vector ===> YES  NO or YES
   Length of Key  ===> ________________      For AES:  16, 24, or 32
   Key Values     ===>
     ________________ ,_______________ ,________________,____________________
   Comment Line   ===> _______________________________________________________

Press ENTER to create and store control statement
Press END   to exit to the previous panel without saving
```

Figure 66. Selecting the ADD Function on the Create ADD, UPDATE, or DELETE Key Statement Panel

3. In the Key Type field, enter the type of key you want KGUP to process with the control statement. This field represents the TYPE keyword on the control statement.

   If you leave the Key Type Field blank and press ENTER, the Key Type Selection panel appears. See Figure 67 on page 240.



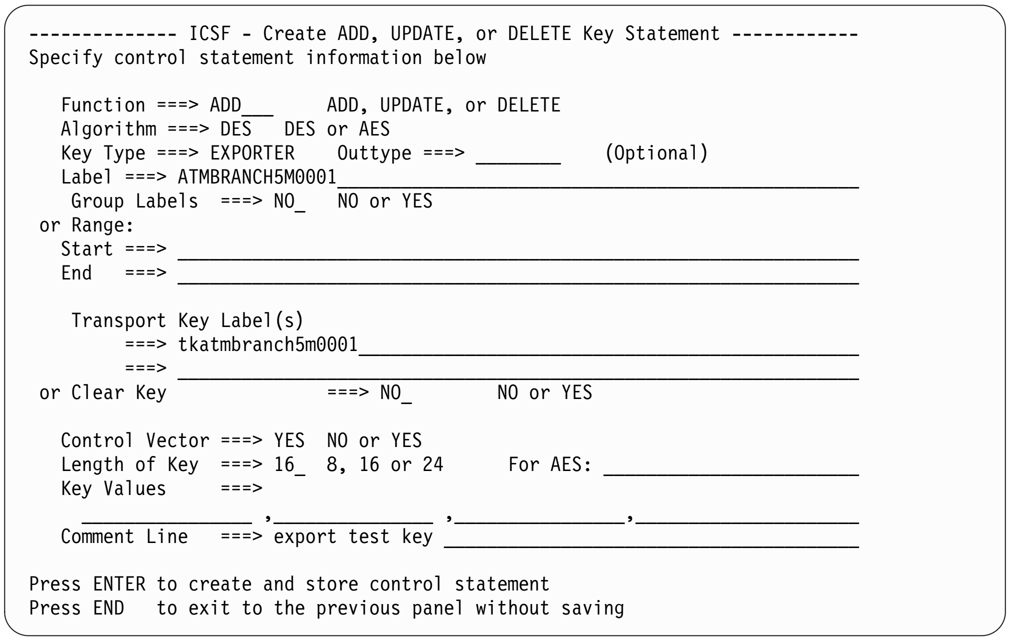
```
CSFCSE12-------- ICSF - Key Type Selection Panel ----  ROW 1 TO 13 OF 11
COMMAND ===>                                                SCROLL ===> PAGE

Select one key type only
    KEY TYPE      DESCRIPTION

   CIPHER     Data encryption/decryption key
   CIPHERXI   Input cipher text transaction key
   CIPHERXL   Cipher text transaction key
   CIPHERXO   Output cipher text transaction key
   CLRAES     Clear AES encryption/decryption key
   CLRDES     Clear DES encryption/decryption key
   DATA       Encryption/Decryption key
   DATAM      Double-length MAC generation key
   DATAMV     Double-length MAC verification key
   DECIPHER   Data decryption key
   DKYGENKY   Diversified key-generating key
   ENCIPHER   Data encryption key
   EXPORTER   Export key encrypting key
   IMPORTER   Import key encrypting key
   IMPPKA     Limited authority key encrypting key
   IPINENC    Input PIN encrypting key
   MAC        MAC generate key
   MACVER     MAC verify key
   NULL       Dummy CKDS records
   OPINENC    Output PIN-encrypting key
   PINGEN     PIN generation key
   PINVER     PIN verification key
********************************BOTTOM OF DATA*********************************
```

Figure 67. Selecting a Key on the Key Type Selection Panel

a. Type s to the left of the key type you want to specify from the displayed list of key types.

   In Figure 67, the exporter key is selected.

b. When you have specified a key type, press ENTER to return to the Create ADD, UPDATE, or DELETE Key Statement panel, as shown in Figure 68 on page 241.



```
------------- ICSF - Create ADD, UPDATE, or DELETE Key Statement ------------
Specify control statement information below

   Function ===> ADD___     ADD, UPDATE, or DELETE
   Algorithm ===> DES   DES or AES
   Key Type ===> EXPORTER    Outtype ===> ________    (Optional)
   Label ===> ATMBRANCH5M0001_________________________________________________
    Group Labels  ===> NO_   NO or YES
 or Range:
   Start ===> ________________________________________________________________
   End   ===> ________________________________________________________________

    Transport Key Label(s)
         ===> tkatmbranch5m0001_______________________________________________
         ===> ________________________________________________________________
 or Clear Key              ===> NO_        NO or YES

   Control Vector ===> YES  NO or YES
   Length of Key  ===> 16_  8, 16 or 24      For AES: ______________________
   Key Values     ===>
     ________________ ,______________ ,_______________,___________________
   Comment Line   ===> export test key _______________________________________

Press ENTER to create and store control statement
Press END   to exit to the previous panel without saving
```

Figure 68. Completing the Create ADD, UPDATE, or DELETE Key Statement Panel

If you abbreviated the control statement function, the function now appears in its full form. The type of key you selected on the Key Type Selection panel appears in the Key Type field.

4. Specify either a label or range to identify the label of the key entry in the CKDS that you want KGUP to process.

   The Label field represents the LABEL keyword on the control statement. The Range field represents the RANGE keyword on the control statement. In the Range fields, specify the first and last label in a range of labels you want KGUP to process.

Table 51. Selecting range and label options

Option	Steps
To have KGUP process only one key label	1. Specify the key label in the Label field. 2. Type `NO` in the Group Labels field.
To have KGUP process more than one key label	1. Specify the first label in the Label field. 2. Type `YES` in the Group Labels field.

5. Specify either a transport key label or `YES` in the Clear Key field.

   The Transport Key Label field represents the TRANSKEY keyword on the control statement. The Clear Key field represents the CLEAR keyword. These keywords are mutually exclusive.

   When KGUP generates a key, the program places the key value in a data set so you can send the value to another system. The other system uses the value to create the complement of the key. You send the key value as either a clear key value or a key value encrypted under a transport key.

   When KGUP imports a key value, the program may import a clear or encrypted key value. KGUP decrypts the encrypted key value from under the transport key that you specify in the Transport Key Label field.

Table 52. Selecting the Transport Key Label and Clear Key Label Options

Option	Steps
To have KGUP generate a key other than an importer key and encrypt the key value	1. Specify the label of the transport key you want KGUP to use to encrypt the key in the Transport Key Label field. 2. Type `NO` in the Clear Key field.
To have KGUP generate a key other than an importer key and leave the key value in the clear	1. Leave the Transport Key Label field blank 2. Type `YES` in the Clear Key field.
To have KGUP import an encrypted key	1. Specify the label of the transport key you want KGUP to use to dencrypt the key in the Transport Key Label field. 2. Type `NO` in the Clear Key field.
To have KGUP import a clear key	1. Leave the Transport Key Label field blank 2. Type `YES` in the Clear Key field.

6. Specify either `YES` or `NO` in the Control Vector field.

   Usually the cryptographic facility exclusive ORs a transport key with a control vector prior to the transport key encrypting a key. However, if your system is exchanging keys with a system like PCF that does not use control vectors, you need to specify that no control vector be used. If you want KGUP to generate a transport key that uses a control vector, type `YES` in the Control Vectors field. Otherwise type `NO`. If you type `NO` in this field, the control statement contains the NOCV keyword.
7. If you want KGUP to work with a single-length key in its processing, type `YES` in the Length of Key field. Otherwise, type `NO`. If you type `YES` in the field, the control statement contains the LENGTH keyword.
8. If you are entering a key value, enter the key value in the Key Values field.

   You enter the value as three values if the key is a triple-length key, two values if the key is a double-length key, or as one value if the key is a single-length key. The Key Values field represents the KEY keyword on the control statement.
9. In the Comment Line field, you can enter up to 45 characters of information about the control statement. The information appears as a comment that precedes the control statement in the input control statement data set.
10. When you enter all the information on this panel, press ENTER.

    If you entered `YES` in the Group Labels field, the Group Label panel appears. See Figure 69 on page 243.

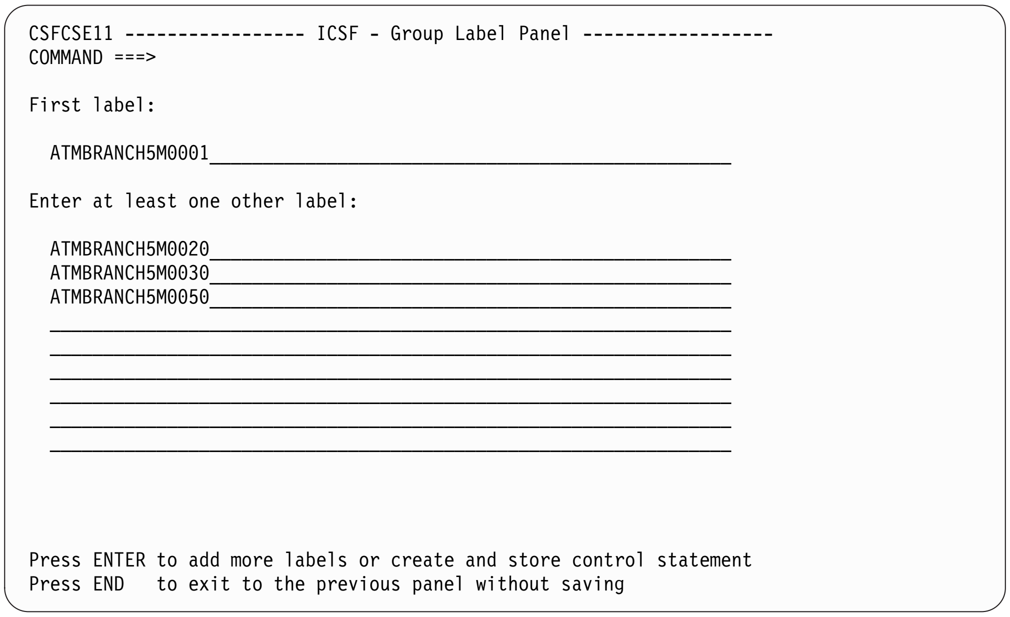

```
CSFCSE11 ----------------- ICSF - Group Label Panel ------------------
COMMAND ===>

First label:

  ATMBRANCH5M0001__________________________________________________

Enter at least one other label:

  ATMBRANCH5M0020__________________________________________________
  ATMBRANCH5M0030__________________________________________________
  ATMBRANCH5M0050__________________________________________________
  _________________________________________________________________
  _________________________________________________________________
  _________________________________________________________________
  _________________________________________________________________
  _________________________________________________________________
  _________________________________________________________________




Press ENTER to add more labels or create and store control statement
Press END   to exit to the previous panel without saving
```

Figure 69. Specifying Multiple Key Labels on the Group Label Panel

   a. Enter any additional key labels you want KGUP to process with the control statement.

      The first label you entered in the Label field of the Create ADD, UPDATE, or DELETE Key Statement panel appears at the top of this panel. If you enter duplicate labels, an error message appears on the right side of the panel and the cursor appears on the duplicate label. If the syntax of the label is incorrect, an error message appears and the cursor appears on the incorrect label.

   b. If you have more labels than will fit on this panel, press the ENTER key when you have filled each line on the panel. An additional Group Label Panel appears. Type the remaining labels and press ENTER.

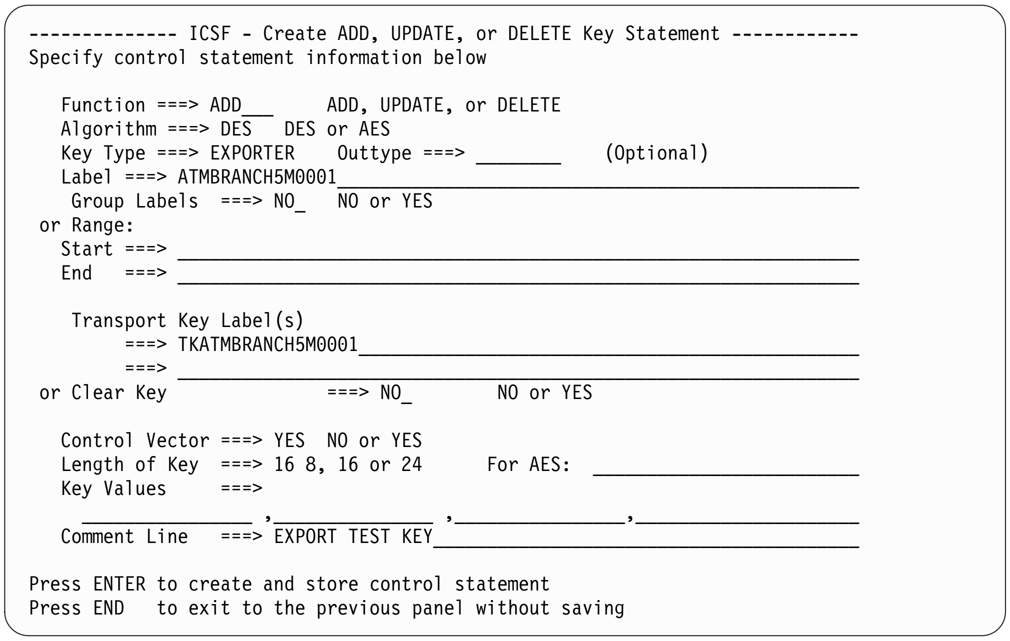
      ICSF writes the control statement to the input control statement data set. You return to the Create ADD, UPDATE, or DELETE Key Statement panel.

   If you entered `NO` in the Group Labels field, you do not access the Group Label panel. You remain on the Create ADD, UPDATE, or DELETE Key Statement panel.

11. Press ENTER to have ICSF write the control statement in the input control statement data set.

    If a specification in any field is incorrect, when ICSF processes the control statement it displays an appropriate message on the top line of the panel. The cursor then appears in the field with the error. To display the long version of the error message at the bottom of the panel, press the HELP key (F1). If you correct the error and press ENTER again, ICSF writes the control statement to the control statement input data set.

    If a control statement was created, the message `SUCCESSFUL UPDATE` appears on the right side of the top line of the panel, as shown in Figure 70 on page 244.



```
-------------- ICSF - Create ADD, UPDATE, or DELETE Key Statement ------------
Specify control statement information below

   Function ===> ADD___     ADD, UPDATE, or DELETE
   Algorithm ===> DES   DES or AES
   Key Type ===> EXPORTER    Outtype ===> ________    (Optional)
   Label ===> ATMBRANCH5M0001________________________________________________
    Group Labels  ===> NO_   NO or YES
 or Range:
   Start ===> ________________________________________________________________
   End   ===> ________________________________________________________________

    Transport Key Label(s)
        ===> TKATMBRANCH5M0001______________________________________________
        ===> ________________________________________________________________
 or Clear Key              ===> NO_        NO or YES

   Control Vector ===> YES  NO or YES
   Length of Key  ===> 16 8, 16 or 24      For AES:  ______________________
   Key Values     ===>
     ________________ ,________________ ,________________,___________________
   Comment Line   ===> EXPORT TEST KEY________________________________________

Press ENTER to create and store control statement
Press END   to exit to the previous panel without saving
```

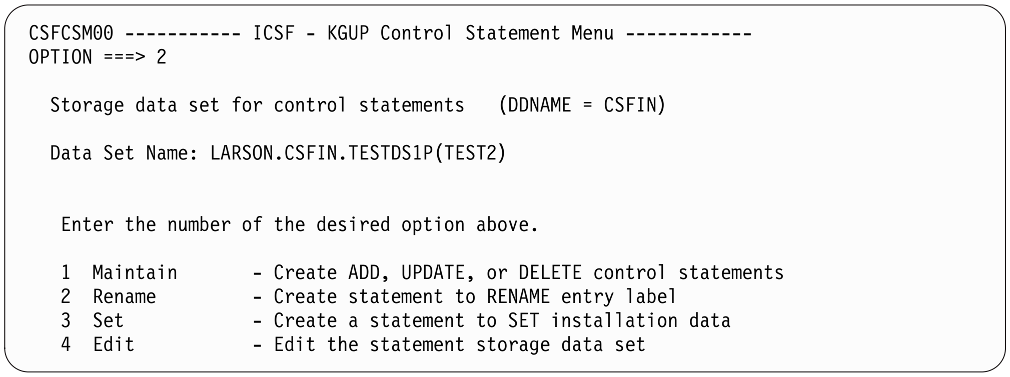
Figure 70. Create ADD, UPDATE, or DELETE Key Statement Panel Showing Successful Update

12. If you want to create another ADD, UPDATE, or DELETE control statement, enter new information in the fields to create the control statement.
13. When you specify the information, press ENTER to place the control statement in the control statement input data set.
14. If you do not want to create another ADD, UPDATE, or DELETE control statement, press END to return to the KGUP Control Statement Menu panel.

Steps for creating a RENAME control statement

The Create RENAME Control Statement panel appears. The RENAME control statement changes the label of a key entry in a CKDS. To create a RENAME control statement:

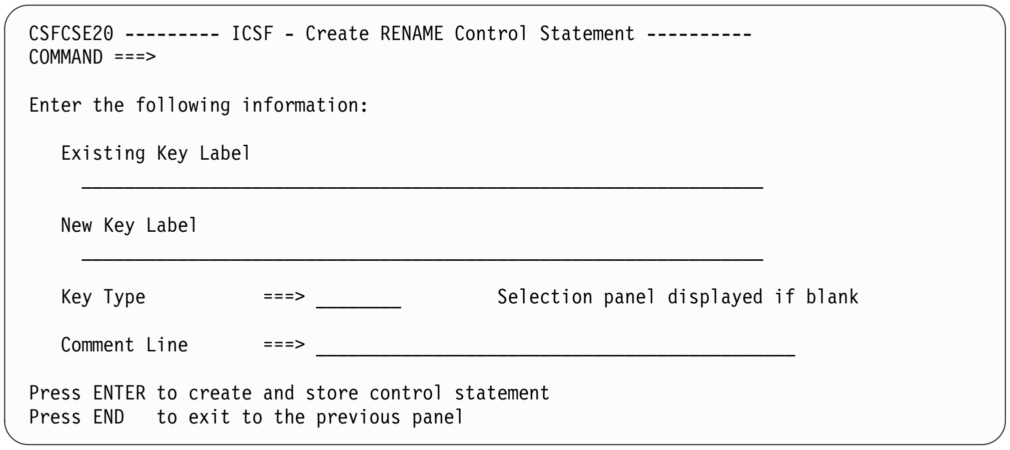
1. Choose option 2 on the KGUP Control Statement Menu, as shown in Figure 71.



```
CSFCSM00 ----------- ICSF - KGUP Control Statement Menu ------------
OPTION ===> 2

  Storage data set for control statements   (DDNAME = CSFIN)

  Data Set Name: LARSON.CSFIN.TESTDS1P(TEST2)


   Enter the number of the desired option above.

   1  Maintain       - Create ADD, UPDATE, or DELETE control statements
   2  Rename         - Create statement to RENAME entry label
   3  Set            - Create a statement to SET installation data
   4  Edit           - Edit the statement storage data set
```

Figure 71. Selecting the Rename Option on the KGUP Control Statement Menu Panel

2. See Figure 72. If you leave this field blank, the On this panel, you enter information in the fields to create a RENAME control statement. This panel creates a RENAME control statement according to the syntax described in "Syntax of the RENAME Control Statement" on page 214. See that topic for more information about the RENAME control statement keywords.



```
CSFCSE20 --------- ICSF - Create RENAME Control Statement ----------
COMMAND ===>

Enter the following information:

   Existing Key Label
     ________________________________________________________________

   New Key Label
     ________________________________________________________________

   Key Type          ===> ________        Selection panel displayed if blank

   Comment Line      ===> _____________________________________________

Press ENTER to create and store control statement
Press END   to exit to the previous panel
```

Figure 72. Create RENAME Control Statement Panel

3. In the Existing Key Label field, specify the current label on the CKDS that you want KGUP to change.
4. In the New Key Label field, specify the new label that you want to replace the existing label.
5. In the Key Type field, specify the key type of the key entry whose label you want changed. Key Type Selection panel appears. See Figure 73 on page 246.

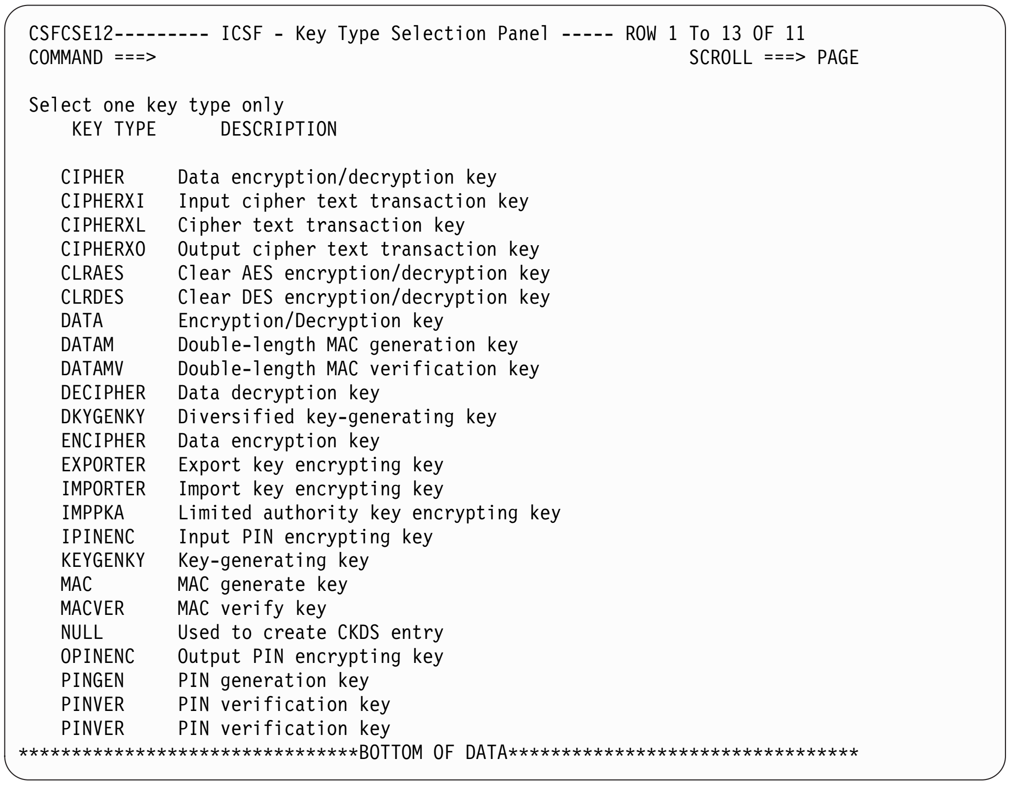

```
CSFCSE12--------- ICSF - Key Type Selection Panel ----- ROW 1 To 13 OF 11
COMMAND ===>                                              SCROLL ===> PAGE

Select one key type only
    KEY TYPE      DESCRIPTION

   CIPHER     Data encryption/decryption key
   CIPHERXI   Input cipher text transaction key
   CIPHERXL   Cipher text transaction key
   CIPHERXO   Output cipher text transaction key
   CLRAES     Clear AES encryption/decryption key
   CLRDES     Clear DES encryption/decryption key
   DATA       Encryption/Decryption key
   DATAM      Double-length MAC generation key
   DATAMV     Double-length MAC verification key
   DECIPHER   Data decryption key
   DKYGENKY   Diversified key-generating key
   ENCIPHER   Data encryption key
   EXPORTER   Export key encrypting key
   IMPORTER   Import key encrypting key
   IMPPKA     Limited authority key encrypting key
   IPINENC    Input PIN encrypting key
   KEYGENKY   Key-generating key
   MAC        MAC generate key
   MACVER     MAC verify key
   NULL       Used to create CKDS entry
   OPINENC    Output PIN encrypting key
   PINGEN     PIN generation key
   PINVER     PIN verification key
   PINVER     PIN verification key
********************************BOTTOM OF DATA*********************************
```

Figure 73. Selecting a Key Type on the Key Type Selection Panel

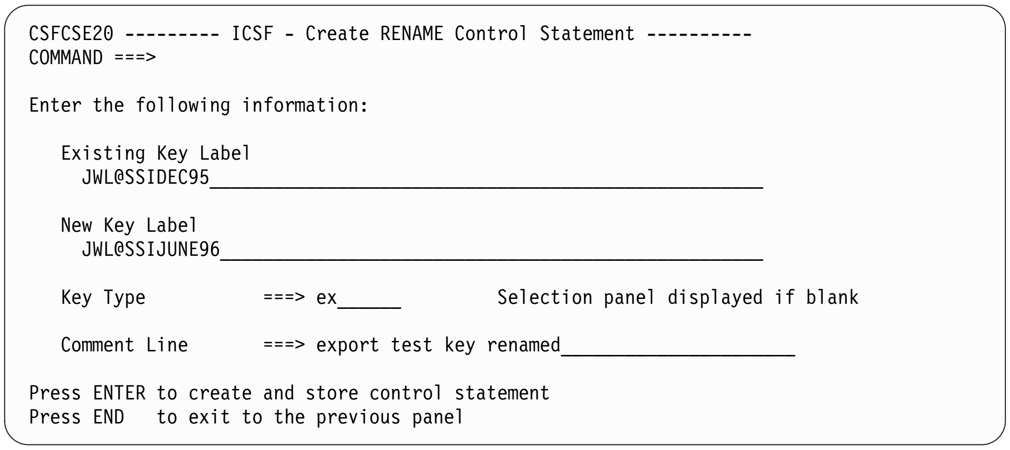
a. Type s to the left of the key type you want to specify.

   In Figure 73, the exporter key is selected.

b. Press ENTER to return to the Create RENAME Control Statement panel.

   The RENAME control statement The key type you choose on the Key Type Selection panel appears in the key type field.

An example of a Create RENAME Control Statement panel which creates a control statement to change the key label JWL@SSIDEC95 to JWL@SSIJUNE96 for an exporter key is shown in Figure 74 on page 247.



```
CSFCSE20 --------- ICSF - Create RENAME Control Statement ----------
COMMAND ===>

Enter the following information:

   Existing Key Label
     JWL@SSIDEC95____________________________________________________

   New Key Label
     JWL@SSIJUNE96___________________________________________________

   Key Type         ===> ex______        Selection panel displayed if blank

   Comment Line     ===> export test key renamed______________________

Press ENTER to create and store control statement
Press END   to exit to the previous panel
```

Figure 74. Completing the Create RENAME Control Statement Panel

6. In the Comment Line field, you can enter up to 45 characters of information about the control statement.

   The information appears as a comment that precedes the control statement in the input control statement data set.
7. When you enter all the information on the Create RENAME Control Statement panel, press ENTER.

   ICSF writes the control statement in the input control statement data set.

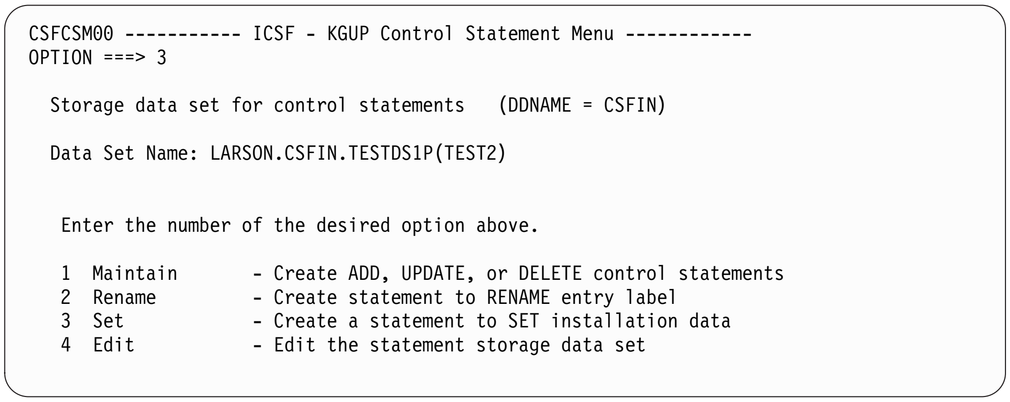
   If a specification in any field is incorrect, when ICSF processes the control statement it displays an appropriate message on the top line of the panel. The cursor then appears in the field with the error. To display the long version of the error message at the bottom of the panel, press the HELP key (F1). You can correct the error and press ENTER again so ICSF can write the control statement to the control statement input data set.

   The Create SET Control Statement panel appears. If a control statement was created, the message `SUCCESSFUL UPDATE` appears on the right side of the top line of the panel.
8. To create another RENAME control statement, enter new information in the fields to create the control statement.
9. When you specify the information, press ENTER to place the control statement in the control statement input data set.
10. When you have finished creating RENAME control statements, press END to return to the KGUP Control Statement Menu panel.

Steps for creating a SET control statement

The SET control statement specifies data for KGUP to send to a KGUP exit routine. To create a SET control statement:

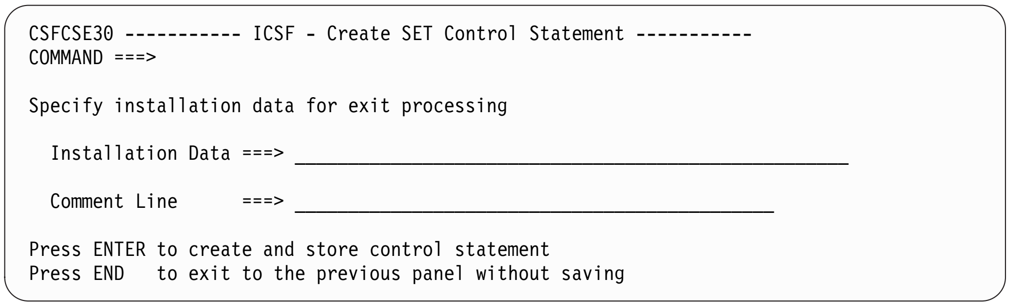
1. Choose option 3 on the KGUP Control Statement Menu, as shown in Figure 75 on page 248.



```
CSFCSM00 ----------- ICSF - KGUP Control Statement Menu ------------
OPTION ===> 3

  Storage data set for control statements   (DDNAME = CSFIN)

  Data Set Name: LARSON.CSFIN.TESTDS1P(TEST2)


   Enter the number of the desired option above.

   1  Maintain       - Create ADD, UPDATE, or DELETE control statements
   2  Rename         - Create statement to RENAME entry label
   3  Set            - Create a statement to SET installation data
   4  Edit           - Edit the statement storage data set
```

Figure 75. Selecting the Set Option on the KGUP Control Statement Menu Panel

2. See Figure 76. From this panel you can creates a SET control statement. For information about the SET control statement keywords, refer to “Syntax of the SET Control Statement” on page 215.



```
CSFCSE30 ----------- ICSF - Create SET Control Statement -----------
COMMAND ===>

Specify installation data for exit processing

  Installation Data ===> ____________________________________________________

  Comment Line      ===> _____________________________________________

Press ENTER to create and store control statement
Press END   to exit to the previous panel without saving
```

Figure 76. Create SET Control Statement Panel

3. In the Installation Data field, enter the data to pass to a KGUP installation exit.
4. In the Comment Line field, you can enter up to 45 characters of information about the control statement.

   The information appears as a comment that precedes the control statement in the input control statement data set.

   An example of a Create SET Control Statement panel which passes date information to the installation exit is shown in Figure 77 on page 249.

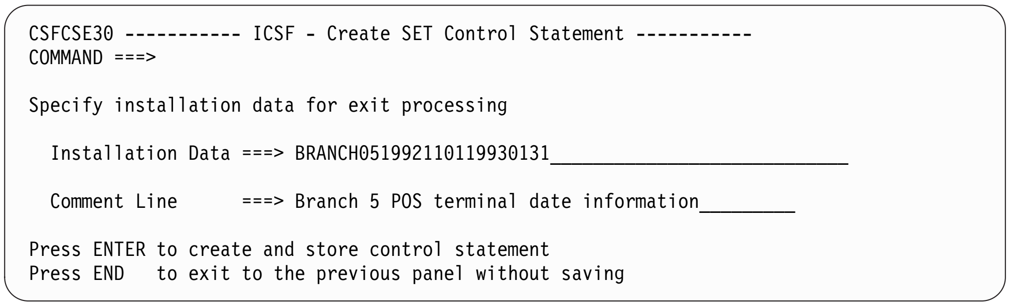

```
CSFCSE30 ----------- ICSF - Create SET Control Statement -----------
COMMAND ===>

Specify installation data for exit processing

  Installation Data ===> BRANCH051992110119930131___________________________

  Comment Line      ===> Branch 5 POS terminal date information_________

Press ENTER to create and store control statement
Press END   to exit to the previous panel without saving
```

Figure 77. Completing the Create SET Control Statement Panel

5. When you enter all the information on this panel, press ENTER.

   ICSF writes the control statement in the input control statement data set.

   When the control statement is created, the message `SUCCESSFUL UPDATE` appears on the right side of the top line of the panel.
6. Press END to return to the KGUP Control Statement Menu panel.

Steps for editing control statements

You can edit the control statement input data set that you specified for this KGUP job. The control statement input data set contains the control statements you created when you specified the control statement input data set.

To edit the control statements you created:

1. Choose option 4 on the KGUP Control Statement Menu panel, as shown in Figure 78.

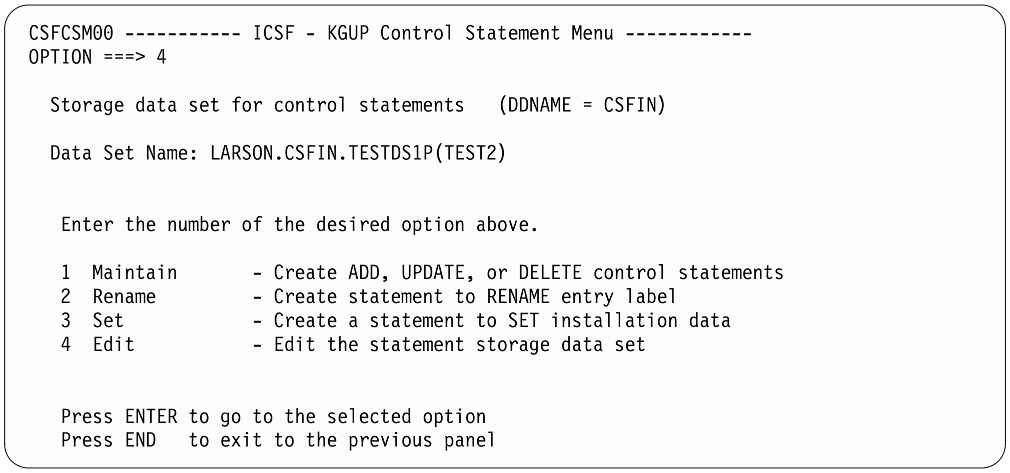
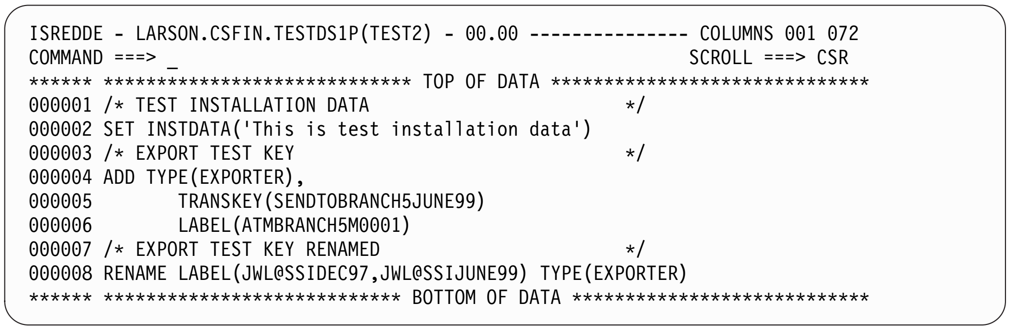

```
CSFCSM00 ----------- ICSF - KGUP Control Statement Menu ------------
OPTION ===> 4

  Storage data set for control statements   (DDNAME = CSFIN)

  Data Set Name: LARSON.CSFIN.TESTDS1P(TEST2)


   Enter the number of the desired option above.

   1  Maintain       - Create ADD, UPDATE, or DELETE control statements
   2  Rename         - Create statement to RENAME entry label
   3  Set            - Create a statement to SET installation data
   4  Edit           - Edit the statement storage data set


   Press ENTER to go to the selected option
   Press END   to exit to the previous panel
```

Figure 78. Selecting the Edit Option on the KGUP Control Statement Menu Panel

   The ISPF editor displays the control statement input data set. An example of a data set called LARSON.CSFIN.TESTDS1P(TEST2) with a SET, ADD, and RENAME control statement is shown in Figure 79 on page 250.



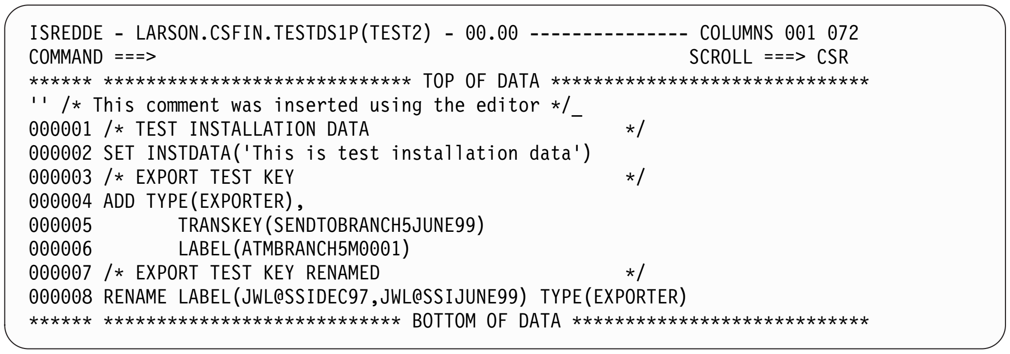
```
ISREDDE - LARSON.CSFIN.TESTDS1P(TEST2) - 00.00 --------------- COLUMNS 001 072
COMMAND ===> _                                                SCROLL ===> CSR
****** ***************************** TOP OF DATA ******************************
000001 /* TEST INSTALLATION DATA                              */
000002 SET INSTDATA('This is test installation data')
000003 /* EXPORT TEST KEY                                     */
000004 ADD TYPE(EXPORTER),
000005        TRANSKEY(SENDTOBRANCH5JUNE99)
000006        LABEL(ATMBRANCH5M0001)
000007 /* EXPORT TEST KEY RENAMED                             */
000008 RENAME LABEL(JWL@SSIDEC97,JWL@SSIJUNE99) TYPE(EXPORTER)
****** **************************** BOTTOM OF DATA ****************************
```

Figure 79. Edit Control Statement Initial Display Panel

2. You can change any information on the control statements in the data set. You can also add lines to the data set that contains comments or control statements.
3. To specify many similar control statements, copy lines in this file and edit them to create additional control statements.

   Note: The panel does not check whether the control statements that you change are syntactically correct.

   Figure 80 shows the insertion of a comment line in the file.



```
ISREDDE - LARSON.CSFIN.TESTDS1P(TEST2) - 00.00 --------------- COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> CSR
****** ***************************** TOP OF DATA ******************************
'' /* This comment was inserted using the editor */_
000001 /* TEST INSTALLATION DATA                              */
000002 SET INSTDATA('This is test installation data')
000003 /* EXPORT TEST KEY                                     */
000004 ADD TYPE(EXPORTER),
000005        TRANSKEY(SENDTOBRANCH5JUNE99)
000006        LABEL(ATMBRANCH5M0001)
000007 /* EXPORT TEST KEY RENAMED                             */
000008 RENAME LABEL(JWL@SSIDEC97,JWL@SSIJUNE99) TYPE(EXPORTER)
****** **************************** BOTTOM OF DATA ****************************
```

Figure 80. Edit Control Statement Data Set with Insert

4. When you make any changes, press END to save the changes and return to the KGUP Control Statement Menu panel.

Steps for specifying data sets using the ICSF panels

When you run a KGUP job, you must specify the KGUP data sets for the program to use in its processing.

1. To access the panels to specify KGUP data sets, select option 2 on the Key Administration panel, as shown in Figure 81 on page 251, and press ENTER.

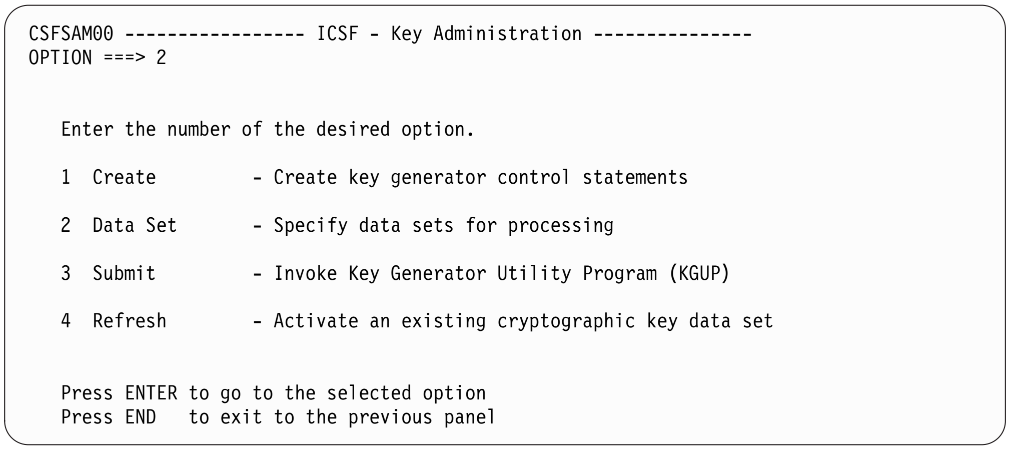

```
CSFSAM00 ----------------- ICSF - Key Administration ---------------
OPTION ===> 2


   Enter the number of the desired option.

   1  Create         - Create key generator control statements

   2  Data Set       - Specify data sets for processing

   3  Submit         - Invoke Key Generator Utility Program (KGUP)

   4  Refresh        - Activate an existing cryptographic key data set


   Press ENTER to go to the selected option
   Press END   to exit to the previous panel
```

Figure 81. Selecting the Specify Data Set Option on the Key Administration Panel

The Specify KGUP Data Sets panel appears. See Figure 82.

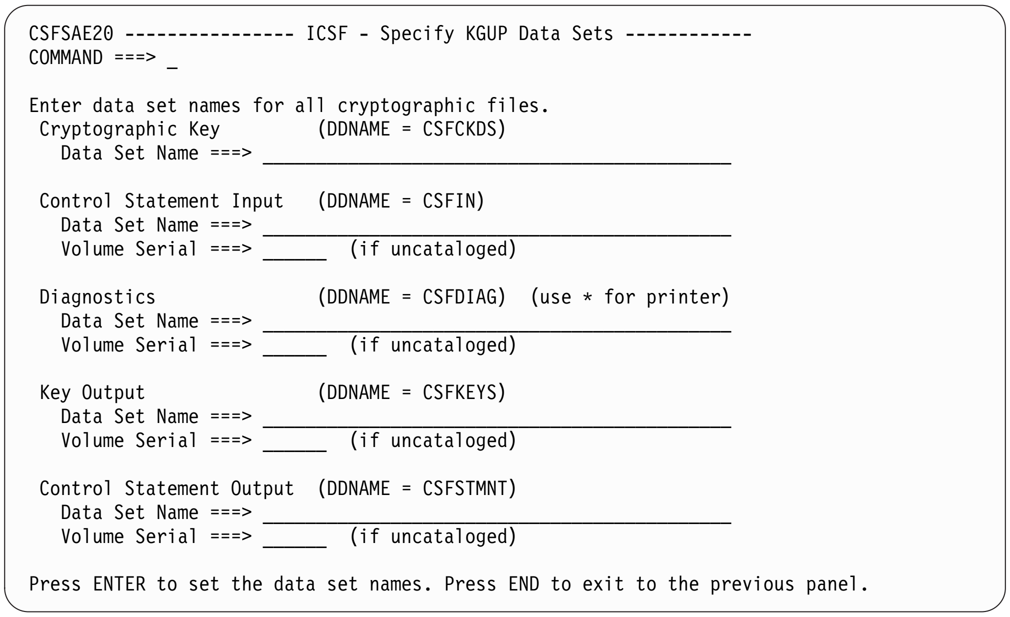

```
CSFSAE20 ---------------- ICSF - Specify KGUP Data Sets ------------
COMMAND ===> _

Enter data set names for all cryptographic files.
 Cryptographic Key         (DDNAME = CSFCKDS)
   Data Set Name ===> ____________________________________________

 Control Statement Input   (DDNAME = CSFIN)
   Data Set Name ===> ____________________________________________
   Volume Serial ===> ______  (if uncataloged)

 Diagnostics               (DDNAME = CSFDIAG)  (use * for printer)
   Data Set Name ===> ____________________________________________
   Volume Serial ===> ______  (if uncataloged)

 Key Output                (DDNAME = CSFKEYS)
   Data Set Name ===> ____________________________________________
   Volume Serial ===> ______  (if uncataloged)

 Control Statement Output  (DDNAME = CSFSTMNT)
   Data Set Name ===> ____________________________________________
   Volume Serial ===> ______  (if uncataloged)

Press ENTER to set the data set names. Press END to exit to the previous panel.
```

Figure 82. Specify KGUP Data Sets Panel

This panel contains all the data sets that KGUP uses for input or output during processing. In the Data Set Name field under each type of data set, you specify the name of the data set for KGUP to use.

2. In the Cryptographic Key Data Set Name field, specify the name of the CKDS which contains the key entries that KGUP processes.

   You must initialize the CKDS by using the method that is described in "Initializing the key data sets at first-time startup" on page 128. The data set can be any disk copy of a CKDS that is enciphered under the current master key.

3. In the Control Statement Input Data Set Name field, specify the name of the data set that contains the control statements you want KGUP to process for this job.

4. In the Volume Serial field, enter the volume serial for the data set if it is not cataloged.

   If you specified a control statement input data set on the KGUP Control Statement Data Set Specification panel, the data set name appears in the Control Statement Input Data Set Name field on this panel. If you change the data set name on this panel, it automatically changes on the KGUP Control Statement Data Set Specification panel. Refer to Figure 60 on page 234 for an example of the KGUP Control Statement Data Set Specification panel.
5. In the Diagnostics Data Set Name field, specify the name of the data set where KGUP places the image of the control statements and any diagnostic KGUP generates.

   You do not have to allocate this data set when you specify the data set in this field. If the data set does not already exist, then a job control language statement that allocates the data set can be used when you submit the job.
6. In the Volume Serial field, enter the volume serial for the data set if the data set already exists but is not cataloged.

   If you enter an * in the Diagnostics Data Set Name field, the information is printed directly to a printer instead of a data set.
7. In the Key Output Data Set Name field, specify the name of the data set that contains key values that are generated to use to create complementary key values.

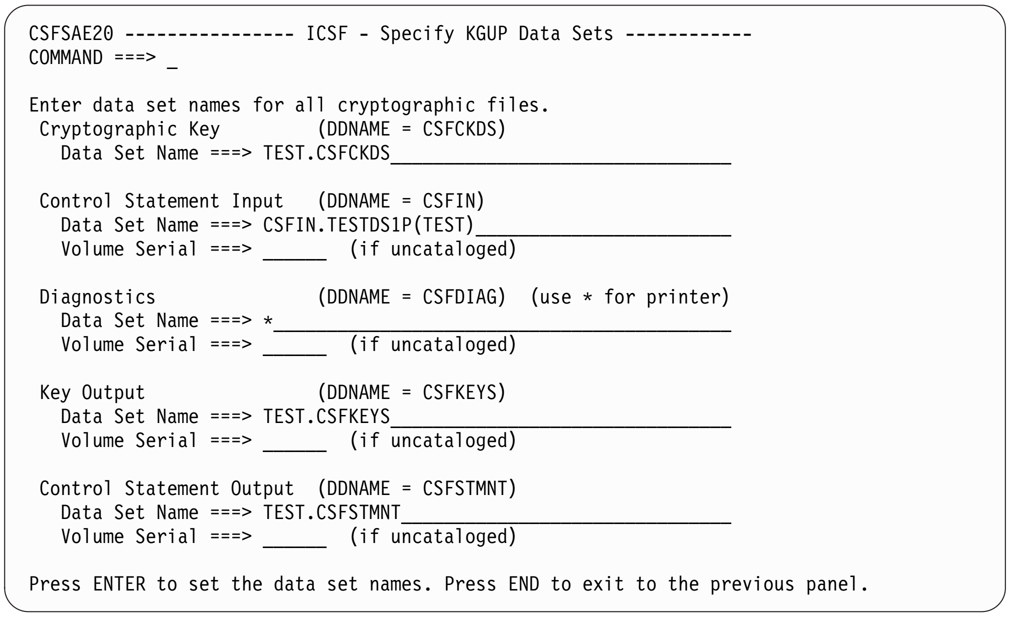
   You do not have to allocate this data set when you specify the data set in this field. If the data set does not already exist, then a job control language statement that allocates the data set can be used when you submit the job.
8. In the Volume Serial field, enter the volume serial for the data set if the data set already exists but is not cataloged.
9. In the Control Statement Output Data Set Name field, specify the name of the data set that contains control statements generated to use to create complementary key values.

   You do not have to allocate this data set when you specify the data set in this field. If the data set does not already exist, then a job control language statement that allocates the data set can be used when you submit the job.
10. In the Volume Serial field, enter the volume serial for the data set if the data set already exists but is not cataloged.

    For a more complete description of each of the data sets, see “Specifying KGUP data sets” on page 224.

    The data sets that you name appear on this panel the next time you access it.

    An example of a Specify KGUP Data Sets panel with the names of data sets specified for KGUP processing is shown in Figure 83 on page 253.



```
CSFSAE20 ---------------- ICSF - Specify KGUP Data Sets ------------
COMMAND ===> _

Enter data set names for all cryptographic files.
 Cryptographic Key         (DDNAME = CSFCKDS)
   Data Set Name ===> TEST.CSFCKDS________________________________

 Control Statement Input   (DDNAME = CSFIN)
   Data Set Name ===> CSFIN.TESTDS1P(TEST)________________________
   Volume Serial ===> ______  (if uncataloged)

 Diagnostics               (DDNAME = CSFDIAG)  (use * for printer)
   Data Set Name ===> *___________________________________________
   Volume Serial ===> ______  (if uncataloged)

 Key Output                (DDNAME = CSFKEYS)
   Data Set Name ===> TEST.CSFKEYS________________________________
   Volume Serial ===> ______  (if uncataloged)

 Control Statement Output  (DDNAME = CSFSTMNT)
   Data Set Name ===> TEST.CSFSTMNT_______________________________
   Volume Serial ===> ______  (if uncataloged)

Press ENTER to set the data set names. Press END to exit to the previous panel.
```

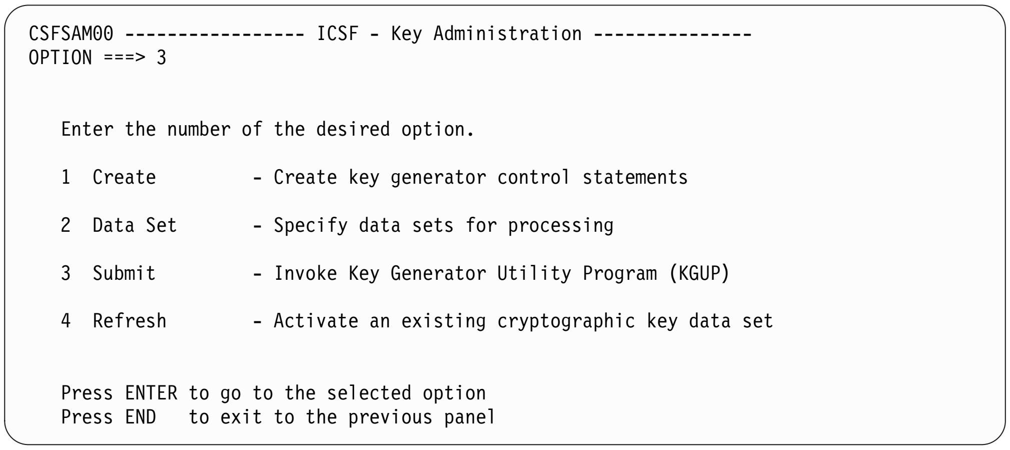
Figure 83. Completing the Specify KGUP Data Sets Panel

11. Press ENTER to set the data set names.
12. Press END to return to the ICSF Key Administration panel.

Steps for creating the job stream using the ICSF panels

The Set KGUP JCL Job Card panel appears. When you create the control statements and specify the data sets for KGUP processing, you submit the job to run KGUP. You submit a KGUP job stream to process control statements which modify a CKDS and output information to other data sets. The names of the data sets that KGUP uses are specified in the job stream.

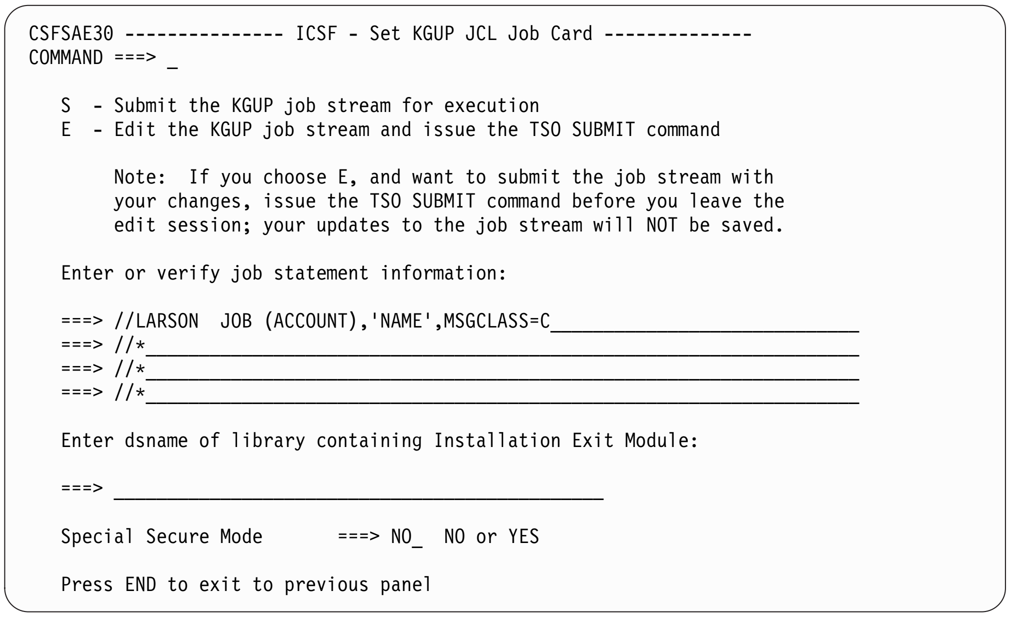
1. To access the panels to create the KGUP job stream, select option 3 on the Key Administration panel, as shown in Figure 84, and press ENTER.



```
CSFSAM00 ----------------- ICSF - Key Administration ---------------
OPTION ===> 3


   Enter the number of the desired option.

   1  Create         - Create key generator control statements

   2  Data Set       - Specify data sets for processing

   3  Submit         - Invoke Key Generator Utility Program (KGUP)

   4  Refresh        - Activate an existing cryptographic key data set


   Press ENTER to go to the selected option
   Press END   to exit to the previous panel
```

Figure 84. Invoking KGUP by Selecting the Submit Option on the Key Administration Panel

See Figure 85 on page 254. The first time you access this panel, the panel displays a JOB statement similar to the one that is shown in this example. ICSF displays your userid as the job name. From this panel you can create a job to run KGUP.



```
CSFSAE30 --------------- ICSF - Set KGUP JCL Job Card --------------
COMMAND ===> _

   S  - Submit the KGUP job stream for execution
   E  - Edit the KGUP job stream and issue the TSO SUBMIT command

        Note:  If you choose E, and want to submit the job stream with
        your changes, issue the TSO SUBMIT command before you leave the
        edit session; your updates to the job stream will NOT be saved.

   Enter or verify job statement information:

   ===> //LARSON  JOB (ACCOUNT),'NAME',MSGCLASS=C____________________________
   ===> //*__________________________________________________________________
   ===> //*__________________________________________________________________
   ===> //*__________________________________________________________________

   Enter dsname of library containing Installation Exit Module:

   ===> ______________________________________________

   Special Secure Mode       ===> NO_  NO or YES

   Press END to exit to previous panel
```

Figure 85. Set KGUP JCL Job Card Panel

2. Change the job statement according to the specifications of your installation.

   The line of the job control language that appears on this panel contains the job card that is needed to submit the job on the Job Entry Subsystem (JES). This panel displays some commonly used parameters that are installation dependent. A job name and the word `JOB` are the only required parameters on a job statement. All the other parameters are only required depending on your installation. You can delete or specify these parameters and add more parameters depending on the requirements of your installation. When you change the information that is displayed, ICSF saves these changes so they appear every time you display the panel.

   a. In the ACCOUNT parameter, enter accounting information as specified by your installation.
   b. In single quotes, enter the name that appears on the output of the job.
   c. In the MSGCLASS parameter, set the output class for the job log.

      When you specify the JOB statement information, the panel displays three comment lines where you can include any information about the job.
   d. If all the parameters do not fit on the first line, delete the * on the second line and continue the JOB statement parameters.
3. If your installation calls an installation exit during KGUP processing and the library containing the exit load module is not in the link list, specify the library in the "Enter dsname of library containing Installation Exit Module" field.

   Because the library must be an authorized library, the library must be defined in your installation's IEAAPFxx member.
4. If any of the control statements contain the CLEAR keyword, specify `YES` in the Special Secure Mode field. Otherwise, ICSF does not have to be in special secure mode, and you should specify `NO` in the Special Secure Mode field.
5. When you specify the necessary information, you can either:

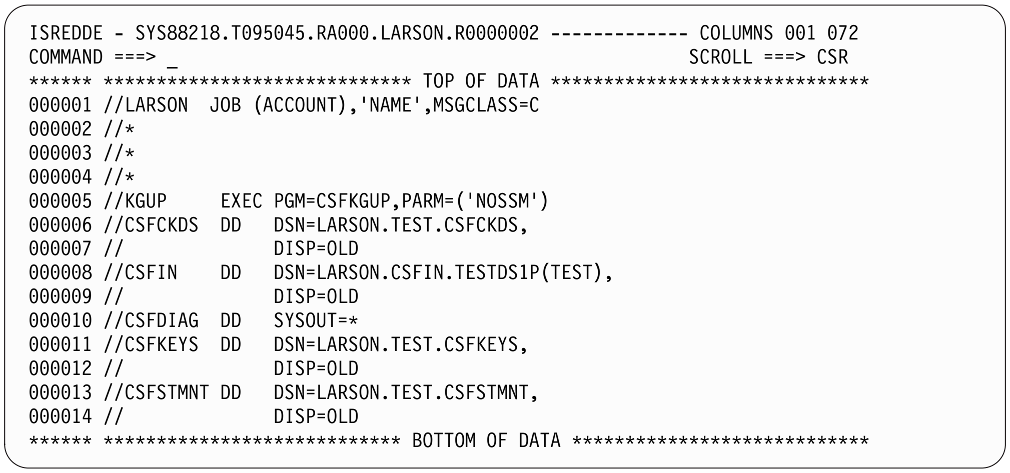
- Enter S to submit the job.

  KGUP creates the job stream and automatically submits the job to run the program.
- Enter E to edit the job.

  KGUP creates the job stream and then displays the job stream on a panel in ISPF edit mode. Figure 86 shows an example of a panel in ISPF edit mode that contains a job stream to run KGUP. When ICSF creates the job stream, ICSF defines the data sets that KGUP uses in the job. It defines these data sets according to the information you specified on the Specify KGUP Data Sets Panel. Refer to Figure 83 on page 253.

  a. On this panel, you can view the job stream ICSF created and make any necessary changes to the job stream.
  b. To submit your job with the changes, you must use the TSO SUBMIT command from the edit session. Type SUBMIT on the command line and press ENTER to submit the job and run KGUP.
  c. To return to the Set KGUP JCL Job Card panel without submitting the job stream, press END.

     The job stream is not saved when you leave this panel.



```
ISREDDE - SYS88218.T095045.RA000.LARSON.R0000002 ------------- COLUMNS 001 072
COMMAND ===> _                                                SCROLL ===> CSR
****** ***************************** TOP OF DATA ******************************
000001 //LARSON  JOB (ACCOUNT),'NAME',MSGCLASS=C
000002 //*
000003 //*
000004 //*
000005 //KGUP     EXEC PGM=CSFKGUP,PARM=('NOSSM')
000006 //CSFCKDS  DD   DSN=LARSON.TEST.CSFCKDS,
000007 //              DISP=OLD
000008 //CSFIN    DD   DSN=LARSON.CSFIN.TESTDS1P(TEST),
000009 //              DISP=OLD
000010 //CSFDIAG  DD   SYSOUT=*
000011 //CSFKEYS  DD   DSN=LARSON.TEST.CSFKEYS,
000012 //              DISP=OLD
000013 //CSFSTMNT DD   DSN=LARSON.TEST.CSFSTMNT,
000014 //              DISP=OLD
****** **************************** BOTTOM OF DATA ****************************
```

Figure 86. KGUP JCL Set for Editing and Submitting (Files Exist)

Example of a KGUP job stream with existing data sets

The KGUP job stream in Figure 86 is an example of a job stream in which the data sets already exist.

In the EXEC statement of the job stream that ICSF created, the PGM parameter specifies that the job run KGUP. The PARM parameter notifies KGUP whether special secure mode is enabled. The keyword SSM indicates that the mode is enabled, and NOSSM indicates that the mode is not enabled.

The data definition (DD) statements identify the data sets that KGUP uses while processing. ICSF uses the names you provide on the Specify KGUP Data Sets panel. The cryptographic key data set (CSFCKDS) and the control statement input data set (CSFIN) have to exist prior to ICSF generating the job stream. The other data sets do not have to already exist. In the example that is shown on this panel, all the data sets existed prior to ICSF creating the job stream.

On the DD statements, the DSN parameter specifies the data set name. ICSF uses the name you provide on the Specify KGUP Data Sets panel for the data set name.

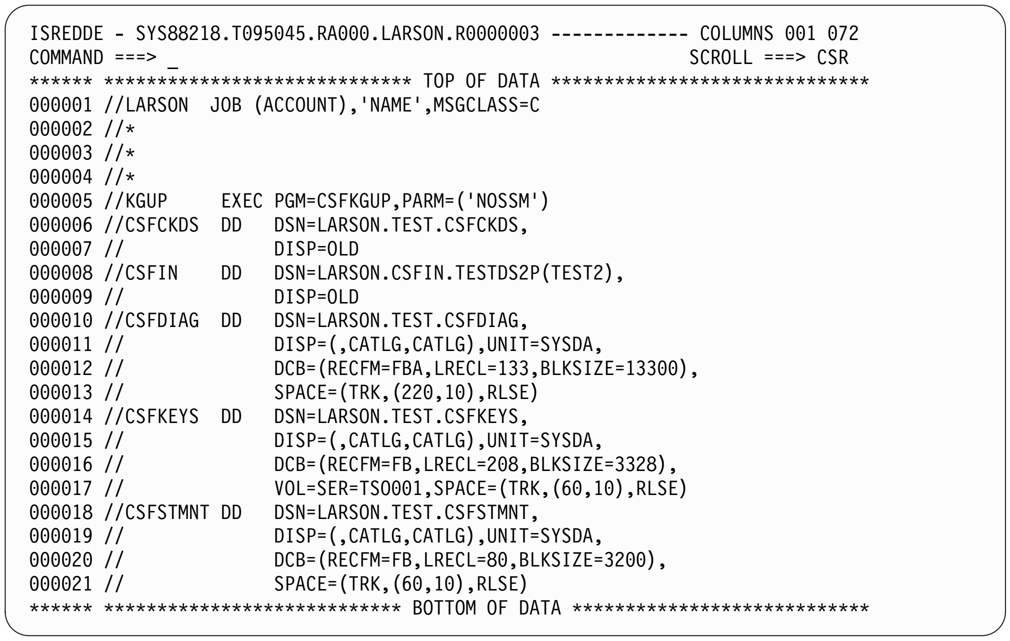
The DISP parameter indicates the data set's status. On this panel, all the data sets existed prior to ICSF creating this job stream, therefore the job stream indicates a status of OLD for the data sets.

In Figure 86 on page 255, the DD statement for the diagnosis data set (CSFDIAG) is different from the other DD statements. The SYSOUT=* parameter specifies that ICSF print the data set on the output listing.

Note: You can change the default values that are used with the job control language such as the record format and record length by changing the outline file, CSFSAJ30. The information appears in the front of CSFSAJ30. CSFSAJ30 resides in the ICSF skeleton library.

Example of a KGUP job stream with non-existing data sets

Figure 87 shows an example of a panel in ISPF edit mode that contains a KGUP job stream where certain data sets did not exist previously.



```
ISREDDE - SYS88218.T095045.RA000.LARSON.R0000003 ------------- COLUMNS 001 072
COMMAND ===> _                                                 SCROLL ===> CSR
****** ****************************** TOP OF DATA ******************************
000001 //LARSON  JOB (ACCOUNT),'NAME',MSGCLASS=C
000002 //*
000003 //*
000004 //*
000005 //KGUP     EXEC PGM=CSFKGUP,PARM=('NOSSM')
000006 //CSFCKDS  DD   DSN=LARSON.TEST.CSFCKDS,
000007 //              DISP=OLD
000008 //CSFIN    DD   DSN=LARSON.CSFIN.TESTDS2P(TEST2),
000009 //              DISP=OLD
000010 //CSFDIAG  DD   DSN=LARSON.TEST.CSFDIAG,
000011 //              DISP=(,CATLG,CATLG),UNIT=SYSDA,
000012 //              DCB=(RECFM=FBA,LRECL=133,BLKSIZE=13300),
000013 //              SPACE=(TRK,(220,10),RLSE)
000014 //CSFKEYS  DD   DSN=LARSON.TEST.CSFKEYS,
000015 //              DISP=(,CATLG,CATLG),UNIT=SYSDA,
000016 //              DCB=(RECFM=FB,LRECL=208,BLKSIZE=3328),
000017 //              VOL=SER=TSO001,SPACE=(TRK,(60,10),RLSE)
000018 //CSFSTMNT DD   DSN=LARSON.TEST.CSFSTMNT,
000019 //              DISP=(,CATLG,CATLG),UNIT=SYSDA,
000020 //              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),
000021 //              SPACE=(TRK,(60,10),RLSE)
****** **************************** BOTTOM OF DATA ****************************
```

Figure 87. KGUP JCL Set for Editing and Submitting (Files Do Not Exist)

The job stream contains information to create the diagnosis data set (CSFDIAG), key output data set (CSFKEYS), and the control statement output data set (CSFSTMNT) that did not previously exist. On the DISP parameter, the CATLG keyword specifies that you want the data set cataloged when the job ends normally and when the job ends abnormally. The unit parameter indicates the device you want the data set to reside on. The DCB parameter specifies the necessary data control block information such as the record format (RECFM), record length (LRECL) and block size (BLKSIZE).

When you submit the job, KGUP performs the functions you specified on the control statements. The functions KGUP performs change the CKDS. You can view the diagnostics data set to know whether KGUP successfully processed the control statements.

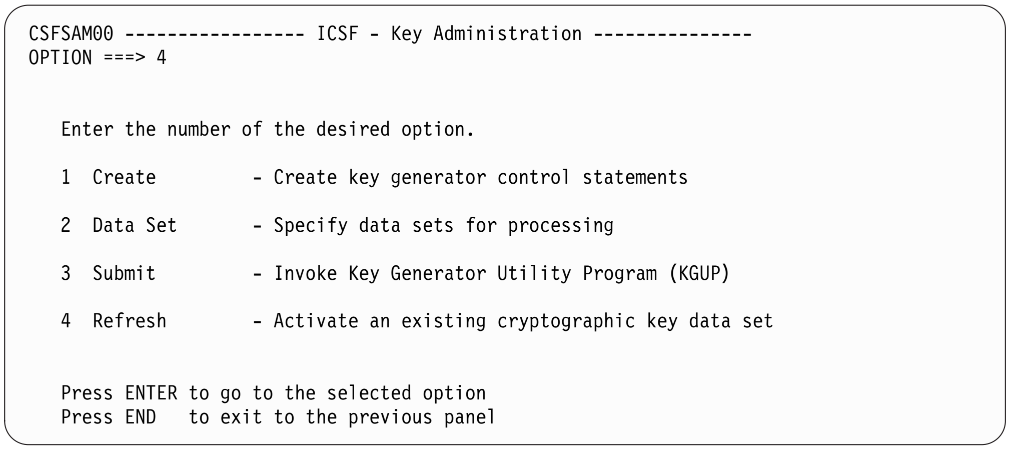
Steps for refreshing the active CKDS using the ICSF panels

KGUP processing affects keys that are stored on a disk copy of the CKDS. You specify the name of the data set when you submit the KGUP job. For information on specifying the disk copy of the CKDS for KGUP processing, see "Steps for specifying data sets using the ICSF panels" on page 250.

ICSF functions use an in-storage copy of the CKDS. To make the changes caused by the KGUP processing active, you replace the in-storage copy of the CKDS with the disk copy that the KGUP processing changed. You refresh the current copy of the CKDS with the changed disk copy of the CKDS. This procedure should be performed on all systems sharing the updated CKDS to ensure they all utilize the updated CKDS records.

Note: The preferred method for performing a CKDS refresh is to use the coordinated refresh function. See "Performing a coordinated CKDS refresh" on page 136 for environment requirements and instructions.

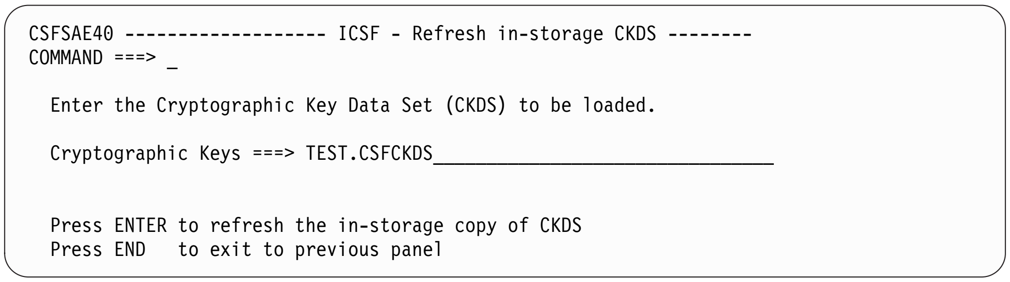
1. To access the panels to refresh the current CKDS, choose option 4 on the Key Administration panel, as shown in Figure 88.



```
CSFSAM00 ----------------- ICSF - Key Administration ---------------
OPTION ===> 4


   Enter the number of the desired option.

   1  Create         - Create key generator control statements

   2  Data Set       - Specify data sets for processing

   3  Submit         - Invoke Key Generator Utility Program (KGUP)

   4  Refresh        - Activate an existing cryptographic key data set


   Press ENTER to go to the selected option
   Press END   to exit to the previous panel
```

Figure 88. Selecting the Refresh Option on the Key Administration Panel

The Refresh in-storage CKDS panel appears. See Figure 89.



```
CSFSAE40 ------------------- ICSF - Refresh in-storage CKDS --------
COMMAND ===> _

  Enter the Cryptographic Key Data Set (CKDS) to be loaded.

  Cryptographic Keys ===> TEST.CSFCKDS________________________________


  Press ENTER to refresh the in-storage copy of CKDS
  Press END   to exit to previous panel
```

Figure 89. Refresh In-Storage CKDS

2. Enter the name of the disk copy of the CKDS to replace the current in-storage copy.

The name of the CKDS that you chose when you specified data sets for KGUP processing on the Specify KGUP Data Sets panel, automatically appears on this panel. If you change the data set name on this panel, the data set name on the Specify KGUP Data Sets panel also changes. Refer to Figure 83 on page 253 for an example of the Specify KGUP Data Sets panel.

3. Press ENTER to replace the in-storage copy of the CKDS with the disk copy.

   Applications that are running on ICSF are not disrupted. A message stating that the CKDS was refreshed appears on the right of the top line on the panel.

   If CKDS record authentication is enabled, ICSF performs a MAC verification on the records when reading the CKDS into storage. If a record fails the MAC verification, the record is not loaded into storage. The operator receives a message indicating the key label and type for that record.
4. Press END to return to the Key Administration Panel.

   Note: If you restart ICSF, the name of the disk copy that you specify in the CKDSN installation option is read into storage.

Scenario of Two ICSF Systems Establishing Initial Transport Keys

This scenario describes how two ICSF systems, System A and System B, establish initial transport keys between themselves. They establish two pairs of complementary importer and exporter keys at each location, as shown in Figure 90.

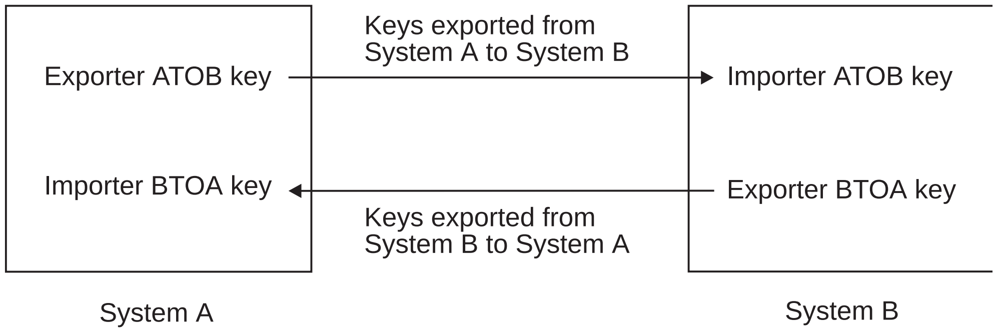


Figure 90. Key Exchange Establishment between Two ICSF Systems

The systems can use these importer and exporter keys during key exchange. First the ICSF administrators at the two locations establish the complementary transport keys to send keys from System A to System B. These keys are the Exporter ATOB key at System A and the Importer ATOB key at System B.

The ICSF administrator at System A submits this control statement to System A's KGUP to create the Exporter ATOB key.

```
ADD  LABEL(ATOB)  TYPE(EXPORTER)  CLEAR
```

KGUP processes this control statement to generate the Exporter ATOB key and places the key in System A's CKDS. KGUP creates a record containing the clear key created for the system, and that record is written to the CSFKEYS data set. This key value must be used to create a control statement like this:

```
ADD  LABEL(ATOB)  TYPE(IMPORTER)  CLEAR,
KEY(B2403EF8125A036F,239AC35A72941EF2)
```

System A can send this control statement to System B, and System B can create the Importer ATOB key. The key value in this control statement is the clear value of the Exporter ATOB key. System A does not send this control statement to System B over the network, because the key value is a clear key value. System A has a courier deliver the control statement to System B.

The administrator at System B submits the control statement to its KGUP. KGUP processes the control statement to create the ATOB importer key. The ATOB exporter key at system A and the ATOB importer key at System B are complementary keys.

This procedure creates a pair of complementary transport keys for keys sent from System A to System B. When System A sends a key to System B it enciphers the key using the ATOB exporter key. When System B receives the key, System B deciphers the key using the ATOB importer key.

Then the ICSF administrators at the two locations establish the complementary transport keys to send keys from System B to System A. These keys are the Importer BTOA key at System A and the Exporter BTOA key at System B.

The ICSF administrator at System A submits this control statement to System A's KGUP to generate the Importer BTOA key.

```
ADD  LABEL(BTOA)  TYPE(IMPORTER)  TRANSKEY(ATOB)
```

KGUP processes this control statement to generate the Importer BTOA key and places the key in System A's CKDS. KGUP also creates this control statement and places the statement in the control statement output data set.

```
ADD  LABEL(BTOA)  TYPE(EXPORTER)  TRANSKEY(ATOB),
KEY(AF04C35A7F1C9636,03CBB854653A0BCF)
```

System A can send this control statement to System B and System B can use the statement to create the Exporter BTOA key. The key value in this control statement is the value of the Importer BTOA key enciphered under the Exporter ATOB key. System A can send this control statement to System B over the network, because the key value is enciphered.

The ICSF administrator at System B submits the control statement to its KGUP. The program processes the control statement to generate the Exporter BTOA key. The Importer BTOA key at System A and the Exporter BTOA key at System B are complementary keys.

This procedure creates a pair of complementary transport keys for keys sent from System B to System A. When System B sends a key to System A, System B enciphers the key using the Exporter BTOA key. When System A receives the key, System A deciphers the key using the Importer BTOA key.

Using these procedures two pairs of complementary transport keys are established at each facility to allow key exchange between the two facilities.

Note:

1. During these procedures, the special secure mode at each system must be enabled, while KGUP is generating or receiving clear key values.
2. The ICSF administrator at System A can submit in the same KGUP job both the ADD control statements meant for processing at System A.

3. The ICSF administrator at System B can submit in the same KGUP job both the ADD control statements meant for processing at System B.

Scenario of an ICSF System and a PCF System Establishing Initial Transport Keys

This scenario describes how an ICSF system and a PCF system establish initial transport keys between themselves. They establish two pairs of complementary importer and exporter keys at each location, as shown in Figure 91.

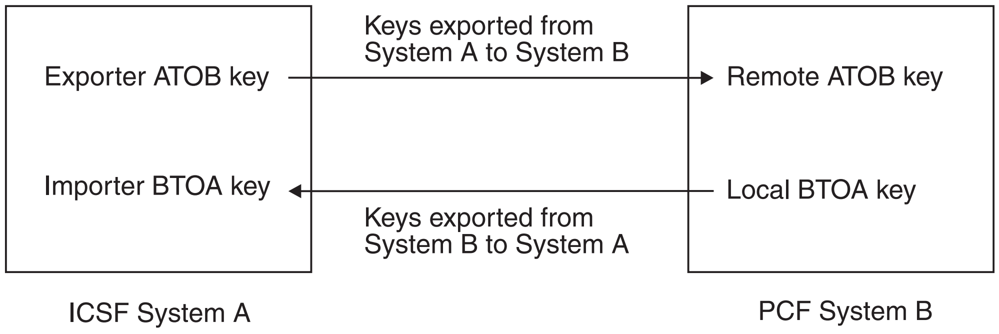


Figure 91. Key Exchange Establishment between an ICSF System and a PCF System

The systems can use these importer and exporter keys during key exchange.

First the ICSF administrators at the two locations establish the complementary transport keys to send keys from ICSF System A to PCF System B. These keys are the Exporter ATOB key at ICSF System A and the Remote ATOB key at PCF System B.

The ICSF administrator at ICSF System A submits this control statement to ICSF System A's KGUP to create the Exporter ATOB key.

```
ADD  LABEL(ATOB)  TYPE(EXPORTER)  CLEAR  NOCV
```

Note: If System B is a PCF system, the ICSF administrator must also specify the keyword SINGLE on this control statement.

KGUP processes this control statement to generate the Exporter ATOB key and places the key in ICSF System A's CKDS. KGUP also creates this control statement and places the statement in the control statement output data set.

```
ADD  LABEL(ATOB)  TYPE(IMPORTER)  CLEAR,
KEY(B2403EF8125A036F,239AC35A72941EF2)  NOCV
```

ICSF System A needs to send this control statement to PCF System B so that PCF System B can create the Remote ATOB key. The key value in this control statement is the clear value of the ATOB exporter key. ICSF System A does not send this control statement to PCF System B over the network, because the key value is a clear key value. ICSF System A has a courier deliver the control statement to System B.

The administrator at either system must change the ICSF control statement format into the PCF control statement format. The administrator could also use information from the key output data set to create the PCF control statement.

The control statement submitted at PCF System B would have this syntax:

```
REMOTE ATOB,KEY=B2403EF8125A036F,IKEY=239AC35A72941EF2,ADD
```

The administrator at PCF System B submits the control statement to the PCF key generation utility program, which processes the control statement to create the ATOB Remote key. The ATOB Exporter key at System A and the ATOB Remote key at PCF System B are complementary keys.

This procedure creates a pair of complementary transport keys for keys sent from ICSF System A to PCF System B. When ICSF System A sends a key to PCF System B, System A enciphers the key using the ATOB exporter key. When PCF System B receives the key, PCF System B deciphers the key using the Remote ATOB key.

Then the ICSF administrators at the two locations establish the complementary transport keys to send keys from PCF System B to ICSF System A. These keys are the Importer BTOA key at ICSF System A and the Local BTOA key at PCF System B.

The ICSF administrator at ICSF System A submits this control statement to ICSF System A's KGUP to generate the Importer BTOA key.

```
ADD  LABEL(BTOA)  TYPE(IMPORTER)  CLEAR  NOCV
```

KGUP processes this control statement to generate the Importer BTOA key and places the statement in ICSF System A's CKDS. KGUP also creates this control statement and places the statement in the control statement output data set.

```
ADD  LABEL(BTOA)  TYPE(EXPORTER)  CLEAR,
KEY(6F3463CA3FBC0626,536B1864954A0B1F)  NOCV
```

System A can send this control statement to System B, which can then use it to create the Local BTOA key. The key value in this control statement is the clear value of the BTOA importer key. ICSF System A does not send this control statement to PCF System B over the network, because the key value is a clear key value. ICSF System A has a courier deliver the control statement to PCF System B.

The administrator at either system must change the ICSF control statement format into the PCF control statement format. The administrator can also use information from the key output data set to create the PCF control statement.

The control statement submitted at PCF System B would have this syntax:

```
LOCAL BTOA,KEY=6F3463CA3FBC0626,IKEY=536B1864954A0B1F,ADD
```

The administrator at PCF System B submits the control statement to the PCF key generation utility program, which processes the control statement to generate the Local BTOA key. The Importer BTOA key at ICSF System A and the Local BTOA key at PCF System B are complementary keys.

Note: A single PCF key generation control statement can be used to generate both Remote and Local BTOA keys, also called a CROSS key pair.

```
CROSS BTOA,KEYLOC=6F3463CA3FBC0626,IKEYLOC=536B1864954A0B1F,
          KEYREM=B2403EF8125A036F,IKEYREM=239AC35A72941EF2,ADD
```

This procedure creates a pair of complementary transport keys for keys sent from PCF System B to ICSF System A. When PCF System B sends a key to ICSF System A, System B enciphers the key, using the Local BTOA key. When ICSF System A receives the key, ICSF System A deciphers the key, using the Importer BTOA key.

By these procedures, two pairs of complementary transport keys are established at each location so that the two systems can exchange keys.

Note: During these procedures, the special secure mode should be enabled while KGUP generates or receives clear key values.

Scenario of an ICSF System and IBM 4765 PCIe and IBM 4764 PCI-X Cryptographic Coprocessors Establishing Initial Transport Keys

This scenario describes how an ICSF system and IBM 4765 PCIe and IBM 4764 PCI-X Cryptographic Coprocessors establish initial transport keys between themselves. They establish two pairs of complementary importer and exporter keys at each location, as shown in Figure 92.

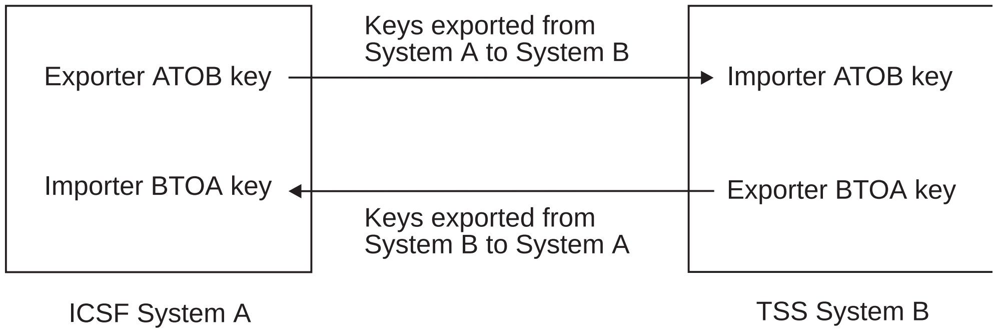


Figure 92. Key Exchange Establishment between IBM 4765 PCIe and IBM 4764 PCI-X Cryptographic Coprocessors systems and an ICSF System

The systems can use these importer and exporter keys during key exchange. First, the ICSF System A administrator and the TSS System B administrator establish the complementary transport keys to send keys from ICSF System A to TSS System B. These keys are the Exporter ATOB key at System A and the Importer ATOB key at System B.

The ICSF administrator at System A submits this control statement to System A's KGUP to create the Exporter ATOB key.

```
ADD  LABEL(ATOB)  TYPE(EXPORTER)  CLEAR
```

KGUP processes this control statement to generate the Exporter ATOB key and places the key in System A's CKDS. KGUP creates a record containing the clear key created for the system, and that record is written to the CSFKEYS data set. ICSF System A then sends this clear key to TSS System B. Because the key value is in the clear, System A has a courier deliver the key, rather than sending it over the network.

The TSS administrator at System B uses the Secure_Key_Import verb to import the ATOB importer key, because the key value is in the clear. The administrator can then use the Key_Record_Create and the Key_Record_Write verbs to place the key in TSS key storage. The ATOB exporter key at ICSF system A and the ATOB importer key at TSS System B are complementary keys.

This procedure creates a pair of complementary transport keys for keys sent from ICSF System A to TSS System B. When ICSF System A sends a key to TSS System B, it enciphers the key using the ATOB exporter key. When TSS System B receives the key, it deciphers the key using the ATOB importer key.

Next, the administrators at the two facilities establish the complementary transport keys to send keys from TSS System B to ICSF System A. These keys are the Importer BTOA key at ICSF System A and the Exporter BTOA key at TSS System B. The ICSF administrator at System A submits this control statement to System A's KGUP to generate the Importer BTOA key.

```
ADD  LABEL(BTOA)  TYPE(IMPORTER)  TRANSKEY(ATOB)
```

KGUP processes this control statement to generate the Importer BTOA key and places the key in System A's CKDS. The ICSF System A administrator can send this key to the TSS System B over the network, because the key value is enciphered.

The TSS administrator at System B uses Key_Import, Key_Record_Create, and the Key_Record_Write verbs to import the key and place it in TSS key storage. The Importer BTOA key at System A and the Exporter BTOA key at System B are complementary keys.

This procedure creates a pair of complementary transport keys for keys sent from TSS System B to ICSF System A. When TSS System B sends a key to ICSF System A, TSS System B enciphers the key using the Exporter BTOA key. When ICSF System A receives the key, it deciphers the key using the Importer BTOA key.

Using these procedures two pairs of complementary transport keys are established at each location to allow key exchange between the two systems.

Note:

1. During these procedures, the special secure mode must be enabled on ICSF while KGUP is generating or receiving clear key values, and the Secure_Key_Import verb must be enabled on TSS to receive clear keys.
2. The ICSF administrator at System A can submit in the same KGUP job both the ADD control statements meant for processing at System A.

Chapter 12. Viewing and Changing System Status

This topic describes:
- "Displaying administrative control functions" on page 266
- "Displaying cryptographic coprocessor status" on page 267
- "Changing coprocessor or accelerator status" on page 269
- "Displaying coprocessor hardware status" on page 270
- "Displaying installation options" on page 279
- "Display CCA domain roles" on page 291
- "Displaying installation exits" on page 302
- "Displaying installation-defined callable services" on page 306

You define installation options, and any installation exits and installation-defined callable services to ICSF. Using the ICSF panels, you can view how these options and programs are currently defined. During master key management, you change the status of the key storage registers that contain key parts and the master keys. You can use the ICSF panels to view the status of these hardware registers. You can also use the ICSF panels to deactivate or activate your cryptographic coprocessors and accelerators.

When you check the status of an installation option, an installation exit, or an installation-defined callable service, you may decide to change how you defined the option or program. You must change the information in the installation options data set and restart ICSF to activate the change.

Identification of cryptographic features

Starting in ICSF FMID HCR77B0, the prefix used to identify Crypto Express2, Crypto Express3, and Crypto Express4 features has changed. Table 1 on page 5 lists the prefix for these features for FMIDs prior to HCR77B0 and the prefix for these features for FMID HCR77B0 and later releases. This change applies to ICSF messages, panels, and publications. The TKE workstation uses this same identification starting with TKE release 8.0.

Table 53. Cryptographic feature identification

Cryptographic feature	Prefix for FMIDs prior to HCR77B0	Prefix for FMID HCR77B0 and later
Crypto Express2 coprocessor	E	2C
Crypto Express2 accelerator	F	2A
Crypto Express3 coprocessor	G	3C
Crypto Express3 accelerator	H	3A
Crypto Express4 CCA coprocessor	SC	4C
Crypto Express4 EP11 coprocessor	SP	4P
Crypto Express4 accelerator	SA	4A

Note: All newer cryptographic features use the convention where *n* is the number in the feature name. For regional cryptographic servers, *n* represents the generation number of the server:

*n***A**	Crypto Express*n* accelerators.
*n***C**	Crypto Express*n* CCA coprocessors.
*n***P**	Crypto Express*n* EP11 coprocessors.
*n***R**	Regional cryptographic servers. Note: 2R requires ICSF FMID HCR77B1 with PTF OA49069 or later.

Displaying administrative control functions

To display administrative control functions:

1. Select option 4, ADMINCNTL, on the "ICSF Primary Menu panel" on page 387. The "CSFACF00 — Administrative Control Functions panel" on page 387 appears.
2. On the "CSFACF00 — Administrative Control Functions panel" on page 387, you can view these options and their values:

   `Dynamic CKDS Access (ENABLED or DISABLED)`
   Specifies whether the dynamic CKDS update services are currently enabled. You can enable or disable these services by placing an 'E' or 'D' for the function on this panel.

Value	Indication
ENABLED	The dynamic CKDS update services are enabled.
DISABLED	The dynamic CKDS update services are disabled.

   `PKA Callable Services (ENABLED or DISABLED)`
   Specifies whether the use of PKA callable services is currently enabled. You can enable or disable these services by placing an 'E' or 'D' for the function on this panel.

Value	Indication
ENABLED	PKA callable services are enabled.
DISABLED	PKA callable services are disabled.

   Note: The PKA callable services control will not appear on the panel if your system has a CEX3C coprocessor.

   `Dynamic PKDS Access (ENABLED or DISABLED)`
   Specifies whether the use of Dynamic PKDS Access callable services are currently enabled. You can enable or disable these services by by placing an 'E' or 'D' for the function on this panel.

Value	Indication
ENABLED	The Dynamic PKDS Access callable services are enabled.
DISABLED	The Dynamic PKDS Access callable services are enabled.

Notes:

- Access to the functions performed using this panel can be controlled by setting up profiles in the CSFSERV class for both CSFRSWS and CSFSSWS.
- If your system is running ICSF FMID HCR77B1 or later, the DISPLAY ICSF,KDS operator command can also be used to show the access state (ENABLED or DISABLED) for a particular KDS. Use the SETICSF command to change the access state if required. For additional information on ICSF operator commands, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Displaying cryptographic coprocessor status

Use the ICSF panels to view the status of the coprocessors. To display coprocessor status:

1. Select option 1, COPROCESSOR MGMT, on the "ICSF Primary Menu panel" on page 387.
2. The "CSFCMP00 — Coprocessor Management panel" on page 390 appears.

On this panel, you can view these options and their values:

`Crypto Feature`
The prefix indicates the type of cryptographic coprocessor or accelerator.

The prefix	Represents a
A	PCI Cryptographic Accelerator
2C	Crypto Express2 Coprocessor
2A	Crypto Express2 Accelerator
3C	Crypto Express3 Coprocessor
3A	Crypto Express3 Accelerator
4A	Crypto Express4 Accelerator
4C	Crypto Express4 CCA Coprocessor
4P	Crypto Express4 PKCS #11 Coprocessor
5A	Crypto Express5 Accelerator
5C	Crypto Express5 CCA Coprocessor
5P	Crypto Express5 PKCS #11 Coprocessor
X	PCI X Cryptographic Coprocessor

`Serial Number`
The serial number is a number assigned to the Crypto Express coprocessors during manufacture. It is displayed for coprocessors configured for CCA or PKCS #11. It is not displayed for accelerators.

`Status`
This field displays the status of the coprocessor.

State Indication

Active (Coprocessors)
All active master keys are correctly set on the coprocessor making the coprocessor usable.

Active (Accelerators)
: The accelerator is available for work.

Master key incorrect (Coprocessors)
: The coprocessor has been configured online. However, at least one master key is incorrectly set or uninitialized. All master keys with MKVPs in the key data sets must be loaded for the coprocessor to become active.

Offline (All)
: The feature may be physically present but it is not available to the operating system. Either it has never been configured online or it has been configured offline by an operator command from the hardware support element.

  Note: If a feature is configured offline from the Support Element, this status display will not be updated automatically. Users will need to pressENTER on this panel to get the latest status.

Disabled by TKE (Coprocessors)
: The feature has been removed from service by the TKE workstation.

Deactivated (All)
: The feature has been deactivated from the Coprocessor Management panel or system console.

Busy (All)
: An unexpected error has been returned from the card. The system goes into recovery to try to reset the card. If the reset is successful, the card is usable again. The user will have to press ENTER to refresh the status on the panel.

Being reconfigured (All)
: An error has been detected and the ICSF configuration task has been invoked to check the feature. The feature may become active if the error is resolved.

Initializing stage 1 (All)
: A newly online feature has been detected by ICSF and ICSF is starting the initialization process. No status is available.

Initializing stage 2 (All)
: A newly online feature or active feature is being reset by ICSF as part of the initialization process or recovery process. No status is available.

Initializing stage 3 (All)
: A newly online feature or inactive feature is being readied to process requests. No status is available.

Hung User on latch (All)
: A feature is not responding and the configuration task is attempting to obtain the feature latch so the feature can be reset. One or more users hold the latch.

Bad feature response (All)
: An unexpected response was received from a feature. The feature is unusable.

Retry limit reached (All)
: While initializing a feature, the limit of attempts to gather status/information was reached. The feature is unusable. ICSF will try again to acquire status.

Unknown response (Coprocessors)
: The coprocessor has returned an unrecognizable code in response to an attempt to determine its status.

Unknown feature type (All)
: A feature has a type that is not recognized by ICSF. The feature is unusable.

AES DES ECC RSA P11
: The state of the master keys in the coprocessor. The state can be U (uninitialized - the current master key register is empty), C (correct - the current master key matches the MKVP in the key data set but the master key is not active), A (active – the master key is active and requests using this master key will be processed by the coprocessor), E (error - the current master key do not match the MKVP in the key data set), or I (ignored – the MKVP is not in the key data set). A hyphen (-) in the state area indicates the key type is not supported.

Note: If your system is running ICSF FMID HCR77B1 or later, the DISPLAY ICSF,CARDS operator command can also be used to show the state of cryptographic coprocessors and accelerators. For additional information on ICSF operator commands, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Changing coprocessor or accelerator status

You can change the status of your cryptographic coprocessors and accelerators, either activating or deactivating them. From the “ICSF Primary Menu panel” on page 387, select option 1, COPROCESSOR MGMT, and the “CSFCMP00 — Coprocessor Management panel” on page 390is displayed.

There are action characters that can be entered on the left of the PCI coprocessor or accelerator number.

Character
: **Indication**

D
: Makes a coprocessor or accelerator unavailable. The status becomes DEACTIVATED. When the request is made, the status of the coprocessor or accelerator may be anything except OFFLINE.

A
: Makes available a coprocessor or accelerator previously deactivated by a 'D' action character.

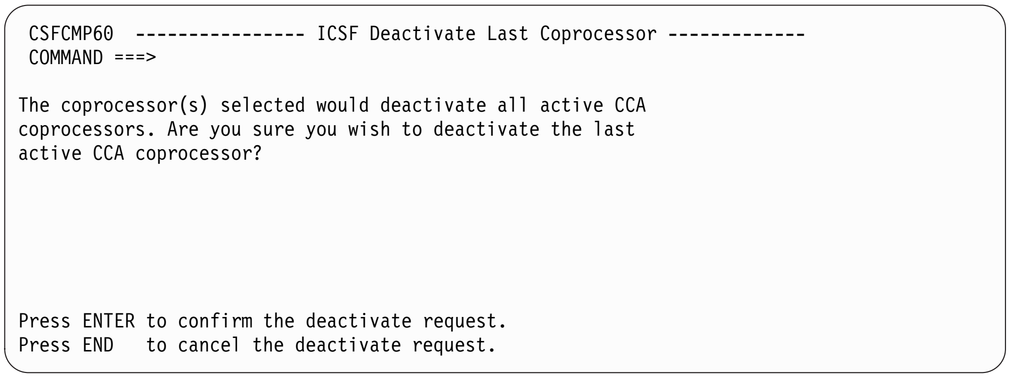
  For a coprocessor, if the coprocessor is online and the master keys are correct, the status will be ACTIVE when the request is made. If the master keys are incorrect, the state will be MasterKeys incorrect.

  For an accelerator, the status will be ACTIVE when the request completes successfully.

Note: If your system is running ICSF FMID HCR77B1 or later, the SETICSF ACTivate and SETICSF DEACTivate operator commands can also be used to change the state of a cryptographic coprocessor or accelerator. For additional information on ICSF operator commands, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

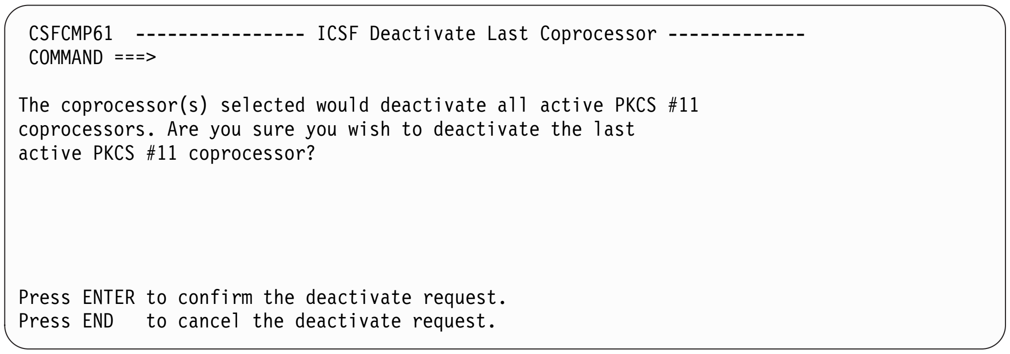
Deactivating the last coprocessor

If there are no CCA cryptographic coprocessors or PKCS #11 coprocessors active, most callable services will fail and most TSO panel utilities will be unavailable. To prevent deactivating the last coprocessor by accident, this panel appears:



```
 CSFCMP60  ---------------- ICSF Deactivate Last Coprocessor -------------
 COMMAND ===>

The coprocessor(s) selected would deactivate all active CCA
coprocessors. Are you sure you wish to deactivate the last
active CCA coprocessor?







Press ENTER to confirm the deactivate request.
Press END   to cancel the deactivate request.
```

Figure 93. Coprocessor Management Panel



```
 CSFCMP61  ---------------- ICSF Deactivate Last Coprocessor -------------
 COMMAND ===>

The coprocessor(s) selected would deactivate all active PKCS #11
coprocessors. Are you sure you wish to deactivate the last
active PKCS #11 coprocessor?







Press ENTER to confirm the deactivate request.
Press END   to cancel the deactivate request.
```

Figure 94. Coprocessor Management Panel

Displaying coprocessor hardware status

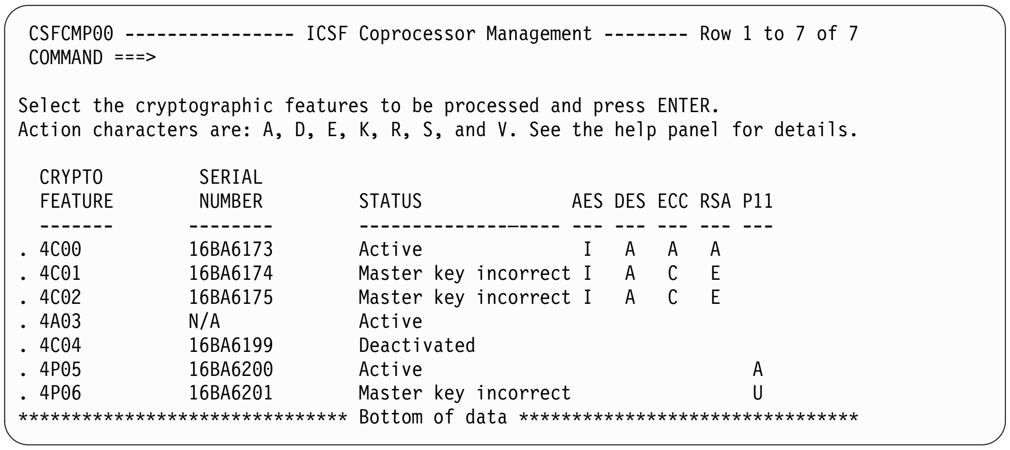
You can use the ICSF panels to view the status of the cryptographic coprocessor key registers, the master key verification patterns, and other information about the cryptographic hardware. You can use this information for master key management.

When you enter and activate an AES, DES, ECC or RSA master key, you change the status of the registers. The cryptographic facility contains three key registers: one for the old master key, one for the new, and one for the current. The current master key register contains the active master key. The old master key is not lost when a new master key is loaded.

Note: The master key verification and hash patterns are displayed as hexadecimal digit strings on the Hardware Status panel. The number of valid digits is determined by the MASTERKCVLEN options data set keyword. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for additional information.

To display coprocessor hardware status:

1. From the Coprocessor Management panel, select the coprocessors to be processed by typing an 'S'.



```
 CSFCMP00 ---------------- ICSF Coprocessor Management -------- Row 1 to 7 of 7
 COMMAND ===>

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

  CRYPTO         SERIAL
  FEATURE        NUMBER         STATUS              AES DES ECC RSA P11
  -------       --------        ------------------- --- --- --- --- ---
. 4C00          16BA6173        Active               I   A   A   A
. 4C01          16BA6174        Master key incorrect I   A   C   E
. 4C02          16BA6175        Master key incorrect I   A   C   E
. 4A03          N/A             Active
. 4C04          16BA6199        Deactivated
. 4P05          16BA6200        Active                               A
. 4P06          16BA6201        Master key incorrect                 U
******************************* Bottom of data ********************************
```

Figure 95. Selecting the coprocessor on the Coprocessor Management Panel

2. Depending on the coprocessor type, one of two different Hardware Status panels appears. Panel CSFCMP40 is displayed for CCA coprocessors (Figure 96 on page 272). When more than two coprocessors are requested, the status display can be scrolled down to show the other coprocessors. You can scroll down using PFKey 8 and up using PFKey 7.

```
CSFCMP40 ----------- ICSF - Coprocessor Hardware Status  ----------------
OPTION ===>

                                                      CRYPTO DOMAIN: 8


REGISTER STATUS                  COPROCESSOR 4C02

 Crypto Serial Number          : 16BA6173
 Status                        : ACTIVE
AES Master Key
   New Master Key register     : EMPTY
     Verification pattern      :
   Old Master Key register     : VALID
     Verification pattern      : BF494FF74B86343F
   Current Master Key register : VALID
     Verification pattern      : 2058C870E9D3194F

DES Master Key
   New Master Key register     : EMPTY
     Verification pattern      :
     Hash pattern              :
                               :
   Old Master Key register     : VALID
     Verification pattern      : 1D08F1C67A1B709A
     Hash pattern              : 2B0C723D1AB9C948
                               : E9C9E32E7FF3B7F4
   Current Master Key register : VALID
     Verification pattern      : CA6B408A02371B1D
     Hash pattern              : DF3A50AE35466123
                               : 96EF557E8BD074C1
ECC Master Key
   New Master Key register     : EMPTY
     Verification pattern      :
   Old Master Key register     : VALID
     Verification pattern      : 9999999999999999
   Current Master Key register : VALID
     Verification pattern      : 9999999999999999

RSA Master Key
   New Master Key register     : EMPTY
     Verification pattern      :
                               :
   Old Master Key register     : VALID
     Verification pattern      : EF4C65754B5088C2
                               : 2D03480BC7B952B2
   Current Master Key register : VALID
     Verification pattern      : E83F158521FEEA23
                               : 986CC9483DAFD711
```

Figure 96. Coprocessor Hardware Status Panel

The coprocessor hardware status fields on this panel contain this information:

CRYPTO DOMAIN
This field displays the value that is specified for the DOMAIN keyword in the installation options data set at ICSF startup. This is the domain in which your system is currently working. It specifies which one of several separate sets of master key registers you can currently access. A system programmer can use the DOMAIN keyword in the installation options data set to specify the domain value to use at ICSF startup. For more information see the DOMAIN installation option.

Crypto Serial Number
The serial number is a number for the 'coprocessor.

Status
This field displays the status of the 'coprocessor.

State	Indication
ACTIVE	The verification pattern for the DES-MK matches the verification pattern of the CKDS. Requests for services can be routed to the coprocessor.
ONLINE	The coprocessor is online. The DES-MK verification pattern does not match the verification pattern in the CKDS. Requests for services cannot be routed to the coprocessor.

DES Master Key

New Master Key Register
This field shows the state of the DES new master key register.

This key register can be in any of these states:

State	Indication
EMPTY	You have not entered any key parts for the initial master key, or you have just transferred the contents of this register into the master key register. Or you have RESET the registers. Or you have zeroized the domain from a TKE workstation or the Support Element.
PART FULL	You have entered one or more key parts but not the final key part.
FULL	You have entered an entire new master key, but have not transferred it to the master key register yet.

Verification Pattern
When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the NMK verification patterns for each unit should match, because the patterns verify the same key.

Hash Pattern
If the master key register is not EMPTY, the panel displays a hash pattern for the key. When you enter a new master key, record the hash pattern that appears on the panel. When the master key becomes active, you can compare the hash patterns to ensure that the one you entered and set is in the master key register.

If your system is using multiple cryptographic coprocessors, you enter the same master key into all units. If the status of the new master key registers are valid, the master key register hash patterns for each unit should match, because the patterns verify the same key.

Old Master Key register

This field shows the states of the DES old master key register.

State	Indication
EMPTY	You have never changed the master key and, therefore, never transferred a master key to the old master key register. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have changed the master key. The master key that was current when you changed the master key was placed in the old master key register.

`Verification Pattern`

When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the DES-MK verification patterns for each unit should match, because the patterns verify the same key.

`Hash Pattern`

If the master key register is not EMPTY, the panel displays a hash pattern for the key. When you enter a new master key, record the hash pattern that appears on the panel. When the master key becomes active, you can compare the hash patterns to ensure that the one you entered and set is in the master key register.

If your system is using multiple cryptographic coprocessors, you enter the same master key into all units. If the status of the new master key registers are valid, the master key register hash patterns for each unit should match, because the patterns verify the same key.

`Current Master Key register`

This field shows the states of the DES master key register.

State	Indication
EMPTY	You have never entered and set an initial symmetric master key. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have entered a new symmetric master key on this coprocessor and chosen either the set or change option.

`Verification Pattern`

When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the NMK verification patterns for each unit should match, because the patterns verify the same key.

Hash Pattern

If the master key register is not EMPTY, the panel displays a hash pattern for the key. When you enter a new master key, record the hash pattern that appears on the panel. When the master key becomes active, you can compare the hash patterns to ensure that the one you entered and set is in the master key register.

If your system is using multiple cryptographic coprocessors, you enter the same master key into all units. If the status of the new master key registers are valid, the master key register hash patterns for each unit should match, because the patterns verify the same key.

AES Master Key

New Master Key Register

This field shows the state of the new master key register.

This key register can be in any of these states:

State	Indication
EMPTY	You have not entered any key parts for the initial master key, or you have just transferred the contents of this register into the master key register. Or you have RESET the registers. Or you have zeroized the domain from a TKE workstation or the Support Element.
PART FULL	You have entered one or more key parts but not the final key part.
FULL	You have entered an entire new master key, but have not transferred it to the master key register yet.

Verification Pattern

When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the NMK verification patterns for each unit should match, because the patterns verify the same key.

Old Master Key register

This field shows the states of the AES old master key register.

State	Indication
EMPTY	You have never changed the master key and, therefore, never transferred a master key to the old master key register. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have changed the master key. The master key that was current when you changed the master key was placed in the old master key register.

Verification Pattern

When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the AES-MK verification patterns for each unit should match, because the patterns verify the same key.

Current Master Key register

This field shows the states of the AES master key register.

State	Indication
EMPTY	You have never entered and set an initial symmetric master key. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have entered a new symmetric master key on this coprocessor and chosen either the set or change option.

Verification Pattern

When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the NMK verification patterns for each unit should match, because the patterns verify the same key.

ECC Master Key

New Master Key Register

This field shows the state of the new master key register.

This key register can be in any of these states:

State	Indication
EMPTY	You have not entered any key parts for the initial master key, or you have just transferred the contents of this register into the master key register. Or you have RESET the registers. Or you have zeroized the domain from a TKE workstation or the Support Element.
PART FULL	You have entered one or more key parts but not the final key part.
FULL	You have entered an entire new master key, but have not transferred it to the master key register yet.

For the CEX2C or CEX3C, there can be an old, new and current master key.

Verification Pattern
When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the NMK verification patterns for each unit should match, because the patterns verify the same key.

Old Master Key register

This field shows the states of the ECC old master key register.

State **Indication**

EMPTY
You have never changed the master key and, therefore, never transferred a master key to the old master key register. Or you have zeroized the domain from a TKE workstation or the Support Element.

VALID
You have changed the master key. The master key that was current when you changed the master key was placed in the old master key register.

Verification Pattern
When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the ECC-MK verification patterns for each unit should match, because the patterns verify the same key.

Current Master Key register
This field shows the states of the ECC master key register.

State **Indication**

EMPTY
You have never entered and set an initial symmetric master key. Or you have zeroized the domain from a TKE workstation or the Support Element.

VALID
You have entered a new symmetric master key on this coprocessor and chosen either the set or change option.

Verification Pattern
When you use the master key panels to enter a new master key, *record the verification pattern* that appears for the master key when the final key part has been entered. You can compare the verification pattern you record with this one to ensure that the key entered and the key in the new master key register are the same.

If your system is using multiple cryptographic coprocessors, you must enter the same master key into all units. If the status of the new master key registers are valid, the NMK verification patterns for each unit should match, because the patterns verify the same key.

RSA Master Key

New Master Key register
This field shows the state of the RSA new master key register.

This key register can be in any of these states:

State	Indication
EMPTY	You have not entered any key parts for the initial RSA master key, or you have just transferred the contents of this register into the RSA master key register. Or you have RESET the registers. Or you have zeroized the domain from a TKE workstation or the Support Element.
PART FULL	You have entered one or more key parts but not the final key part.

Verification Pattern
If the master key register is not EMPTY, a verification pattern is displayed.

Old Master Key register
This field shows the state of the RSA old master key register.

State	Indication
EMPTY	You have never changed the RSA master key and, therefore, never transferred an RSA master key to the RSA old master key register. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have changed the RSA master key. The RSA master key that was current when you changed the master key was placed in the RSA old master key register.

Verification Pattern
If the old asymmetric master key register is valid, the panel displays a verification pattern for the RSA old master key.

Current Master Key register
This field shows the states of the RSA master key register.

State	Indication
EMPTY	You have never entered an initial RSA master key on the coprocessor. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have entered a new RSA master key on this coprocessor.

Verification Pattern
If the RSA master key registers are valid, the panel displays a verification pattern for the key. When you enter a new RSA master key, *record the verification pattern* that appears on the panel. When the RSA master key

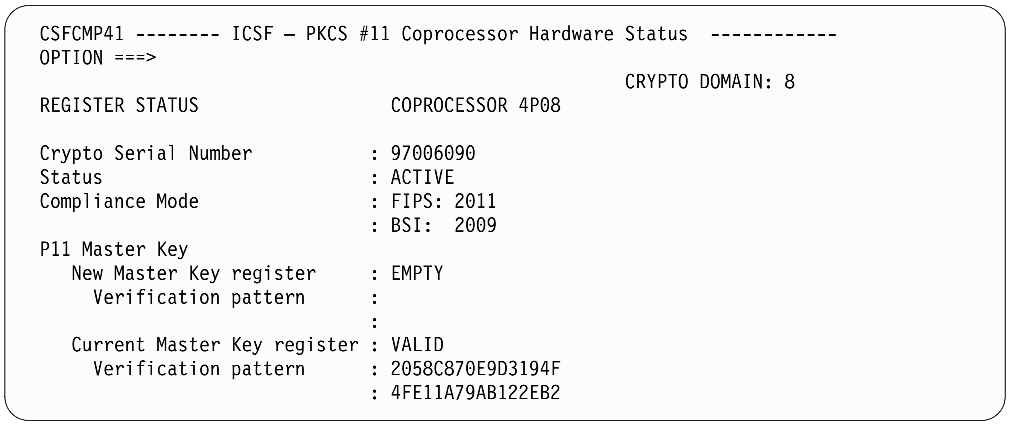
becomes active, you can compare the verification patterns to ensure that the one you entered and set is in the master key register.

The RSA master key must be the same on all the PCI X cards. If the status of all these cryptographic coprocessors is valid, the MK verification patterns for each unit should match, because the patterns verify the same key.

Note: An audit trail of the verification patterns that the CEX2C or CEX3C calculates appears in SMF record type 82.

Panel CSFCMP41 is displayed for Enterprise PKCS #11 coprocessor. Similar to panel CSFCMP40, except that there is only one master key type, the P11 master key, with two registers instead of three:

- The “New Master Key register” - valid states are EMPTY, FULL UNCOMMITTED, or FULL COMMITTED
- The “Current Master Key register” - valid states are EMPTY or VALID



```
CSFCMP41 -------- ICSF – PKCS #11 Coprocessor Hardware Status  ------------
OPTION ===>
                                                       CRYPTO DOMAIN: 8
REGISTER STATUS                  COPROCESSOR 4P08

Crypto Serial Number           : 97006090
Status                         : ACTIVE
Compliance Mode                : FIPS: 2011
                               : BSI:  2009
P11 Master Key
   New Master Key register     : EMPTY
     Verification pattern      :
                               :
   Current Master Key register : VALID
     Verification pattern      : 2058C870E9D3194F
                               : 4FE11A79AB122EB2
```

Figure 97. PKCS #11 Coprocessor Hardware Status Panel

Note: If your system is running ICSF FMID HCR77B1 or later, the DISPLAY ICSF,MKS operator command can also be used to display coprocessor hardware status. For additional information on ICSF operator commands, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

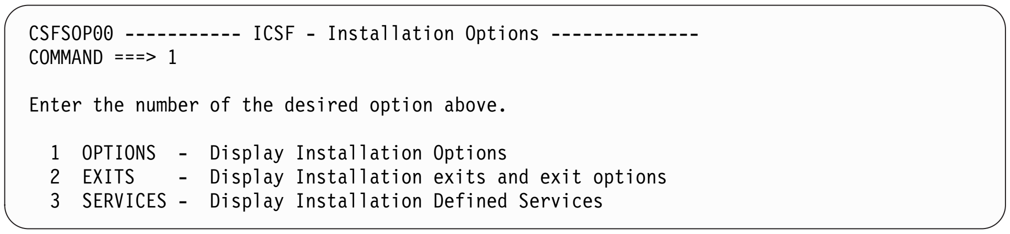
Displaying installation options

Installation options enable you to specify certain modes and conditions to ICSF. For example, if your installation specifies `YES` for the SSM option, you can enable special secure mode. You specify installation options in the installation options data set. The ICSF startup procedure, specifies the installation options data set to be used for that start of ICSF. The options become active, when you start ICSF. You can use the panels to view each installation option and its current value.

To display installation options:

1. Select option 3, OPSTAT, on the “ICSF Primary Menu panel” on page 387.

   The Installation Options panel appears.

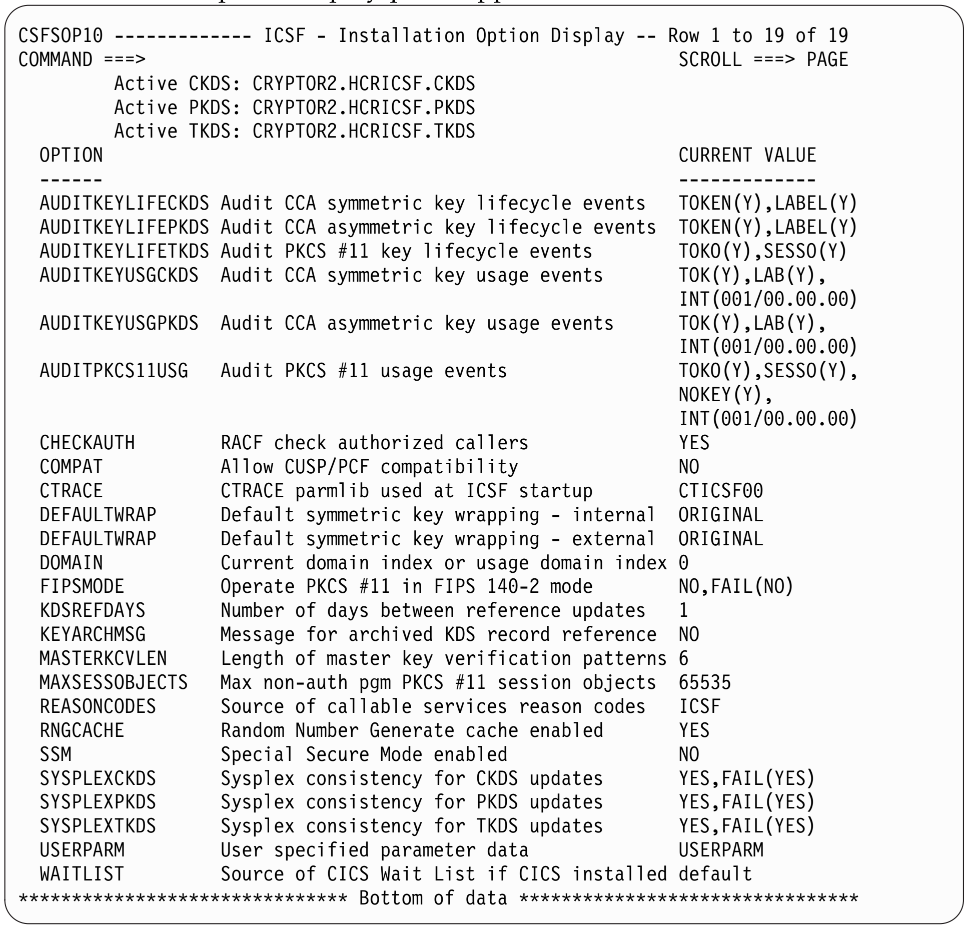


```
CSFSOP00 ----------- ICSF - Installation Options --------------
COMMAND ===> 1

Enter the number of the desired option above.

  1  OPTIONS  -  Display Installation Options
  2  EXITS    -  Display Installation exits and exit options
  3  SERVICES -  Display Installation Defined Services
```

Figure 98. Installation Options panel

2. Select option 1, Options, on the Installation Options panel.

   The Installation Option Display panel appears.



```
CSFSOP10 ------------- ICSF - Installation Option Display -- Row 1 to 19 of 19
COMMAND ===>                                                 SCROLL ===> PAGE
         Active CKDS: CRYPTOR2.HCRICSF.CKDS
         Active PKDS: CRYPTOR2.HCRICSF.PKDS
         Active TKDS: CRYPTOR2.HCRICSF.TKDS
  OPTION                                                     CURRENT VALUE
  ------                                                     -------------
  AUDITKEYLIFECKDS Audit CCA symmetric key lifecycle events   TOKEN(Y),LABEL(Y)
  AUDITKEYLIFEPKDS Audit CCA asymmetric key lifecycle events  TOKEN(Y),LABEL(Y)
  AUDITKEYLIFETKDS Audit PKCS #11 key lifecycle events        TOKO(Y),SESSO(Y)
  AUDITKEYUSGCKDS  Audit CCA symmetric key usage events       TOK(Y),LAB(Y),
                                                              INT(001/00.00.00)
  AUDITKEYUSGPKDS  Audit CCA asymmetric key usage events      TOK(Y),LAB(Y),
                                                              INT(001/00.00.00)
  AUDITPKCS11USG   Audit PKCS #11 usage events                TOKO(Y),SESSO(Y),
                                                              NOKEY(Y),
                                                              INT(001/00.00.00)
  CHECKAUTH        RACF check authorized callers              YES
  COMPAT           Allow CUSP/PCF compatibility               NO
  CTRACE           CTRACE parmlib used at ICSF startup        CTICSF00
  DEFAULTWRAP      Default symmetric key wrapping - internal  ORIGINAL
  DEFAULTWRAP      Default symmetric key wrapping - external  ORIGINAL
  DOMAIN           Current domain index or usage domain index 0
  FIPSMODE         Operate PKCS #11 in FIPS 140-2 mode        NO,FAIL(NO)
  KDSREFDAYS       Number of days between reference updates   1
  KEYARCHMSG       Message for archived KDS record reference  NO
  MASTERKCVLEN     Length of master key verification patterns 6
  MAXSESSOBJECTS   Max non-auth pgm PKCS #11 session objects  65535
  REASONCODES      Source of callable services reason codes   ICSF
  RNGCACHE         Random Number Generate cache enabled       YES
  SSM              Special Secure Mode enabled                NO
  SYSPLEXCKDS      Sysplex consistency for CKDS updates       YES,FAIL(YES)
  SYSPLEXPKDS      Sysplex consistency for PKDS updates       YES,FAIL(YES)
  SYSPLEXTKDS      Sysplex consistency for TKDS updates       YES,FAIL(YES)
  USERPARM         User specified parameter data              USERPARM
  WAITLIST         Source of CICS Wait List if CICS installed default
******************************* Bottom of data ********************************
```

The Installation Options panel displays the keyword for each installation option, a brief description, and the current value of the option.

Note: The REMOTEDEVICE installation options (if specified) are not displayed on this panel. The DISPLAY ICSF operator's console must be used to see the list of remote devices defined. For more information, see the DISPLAY ICSF command in *z/OS Cryptographic Services ICSF System Programmer's Guide*.

You may want to change the current value of an installation option. To change and activate an installation option, you must change the option value in the installation options data set and restart ICSF. A subset of option parameters in the installation options data set are refreshable starting with ICSF FMID HCR77C0. See the

SETICSF command or ICSF Multi-Purpose Service (CSFMPS and CSFMPS6) for details. For integrity reasons, a change of the DOMAIN option also requires a re-IPL of MVS. For a complete description of these installation options and the installation options data set, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

The installation options data set that the system uses at ICSF startup contains keywords and their values which specify certain installation options. On this panel, you can view these options and their values:

`Active CKDS: (data-set-name)`
This specifies the name of the CKDS that ICSF is currently using. During startup, ICSF uses the CKDSN option, but this may be changed by re-enciphering or refreshing the CKDS.

`Active PKDS: (data-set-name)`
This specifies the name of the PKDS that ICSF is currently using. During startup, ICSF uses the PKDSN option, but this may be changed by re-enciphering or refreshing the PKDS.

`Active TKDS: (data-set-name)`
This specifies the name of the TKDS that ICSF is currently using. During startup, ICSF uses the TKDSN option, but this may be changed by re-enciphering or refreshing the TKDS.

`AUDITKEYLIFECKDS(TOKEN(YES or NO),LABEL(YES or NO))`
Provides a set of options that control auditing of events related to the lifecycle of symmetric CCA tokens. The audit logs are in the form of Type 82 SMF records.

`TOKEN(YES or NO)`
Controls lifecycle auditing of CKDS tokens.

Value	Indication
YES	Indicates ICSF should audit lifecycle events related to CKDS tokens. An SMF type 82 subtype 40 record is logged for each event.
NO	No lifecycle auditing of CKDS tokens occurs.

`LABEL(YES or NO)`
Controls lifecycle auditing of CKDS labels.

Value	Indication
YES	Indicates ICSF should audit lifecycle events related to CKDS labels. An SMF type 82 subtype 40 record is logged for each event. The subtype 40 record replaces the subtype 9 record.
NO	No lifecycle auditing of CKDS labels occurs. ICSF continues to log an SMF type 82 subtype 9 record for CKDS updates.

If the AUDITKEYLIFECKDS option is not specified, the default is AUDITKEYLIFECKDS (TOKEN(NO),LABEL(NO)).

Note:

1. An event that involves a token is considered to be any request that uses a token as opposed to a label. This is true regardless of Key Store Policy enablement.

2. If auditing of CKDS labels is enabled, the Key Generator Utility Program (KGUP) needs access to the CSFGKF profile in the CSFSERV class in order to generate the key fingerprint for keys it processes.

For more information about the events that are audited as well as the information contained in the audit record, see Appendix B in *z/OS Cryptographic Services ICSF System Programmer's Guide* for the description for the subtype 40 record.

The auditing of key lifecycle events can also be controlled via the SETICSF operator command. See the description of the SETICSF command in *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

AUDITKEYLIFEPKDS(TOKEN(YES or NO),LABEL(YES or NO))
Provides a set of options that control auditing of events related to the lifecycle of asymmetric CCA tokens. The audit logs are in the form of Type 82 SMF records.

TOKEN(YES or NO)
Controls lifecycle auditing of PKDS tokens.

Value	Indication
YES	Indicates ICSF should audit lifecycle events related to PKDS tokens. An SMF type 82 subtype 41 record is logged for each event.
NO	No lifecycle auditing of PKDS tokens occurs.

LABEL(YES or NO)
Controls lifecycle auditing of PKDS labels.

Value	Indication
YES	Indicates ICSF should audit lifecycle events related to PKDS labels. An SMF type 82 subtype 41 record is logged for each event. The subtype 41 record replaces the subtype 13 record.
NO	No lifecycle auditing of PKDS labels occurs. ICSF continues to log an SMF type 82 subtype 13 record for PKDS updates.

If the AUDITKEYLIFEPKDS option is not specified, the default is AUDITKEYLIFEPKDS (TOKEN(NO),LABEL(NO)).

Note: An event that involves a token is considered to be any request that uses a token as opposed to a label. This is true regardless of Key Store Policy enablement.

For more information about the events that are audited as well as the information contained in the audit record, see Appendix B in *z/OS Cryptographic Services ICSF System Programmer's Guide* for the description for the subtype 41 record.

The auditing of key lifecycle events can also be controlled via the SETICSF operator command. See the description of the SETICSF command in *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

AUDITKEYLIFETKDS(TOKENOBJ(YES or NO),SESSIONOBJ(YES or NO))
Provides a set of options that control auditing of events related to the lifecycle of PKCS #11 objects. The audit logs are in the form of Type 82 SMF records.

TOKENOBJ(YES or NO)
Controls lifecycle auditing of PKCS #11 token objects.

Value Indication

YES Indicates ICSF should audit lifecycle events related to PKCS #11 token objects. An SMF type 82 subtype 42 record is logged for each event. The subtype 42 record replaces the subtype 23 record.

NO No lifecycle auditing of PKCS #11 token objects occurs. ICSF continues to log an SMF type 82 subtype 23 record for TKDS updates.

`SESSIONOBJ(YES or NO)`
Controls lifecycle auditing of PKCS #11 session objects.

Value	Indication
YES	Indicates ICSF should audit lifecycle events related to PKCS #11 session objects. An SMF type 82 subtype 42 record is logged for each event.
NO	No lifecycle auditing of PKCS #11 session objects occurs.

If the AUDITKEYLIFETKDS option is not specified, the default is AUDITKEYLIFETKDS (TOKENOBJ(NO),SESSIONOBJ(NO)).

For more information about the events that are audited as well as the information contained in the audit record, see Appendix B in *z/OS Cryptographic Services ICSF System Programmer's Guide* for the description for the subtype 42 record.

The auditing of key lifecycle events can also be controlled via the SETICSF operator command. See the description of the SETICSF command in *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

`AUDITKEYUSGCKDS(TOKEN(YES or NO),LABEL(YES or NO),INTERVAL(n))`
Provides a set of options that control auditing of events related to the usage of symmetric CCA tokens. The audit logs are in the form of Type 82 SMF records.

`TOKEN(YES or NO)`
Controls usage auditing of CKDS tokens.

Value	Indication
YES	Indicates ICSF should audit usage events related to CKDS tokens. An SMF type 82 subtype 44 record is logged for each event.
NO	No usage auditing of CKDS tokens occurs.

`LABEL(YES or NO)`
Controls usage auditing of CKDS labels.

Value	Indication
YES	Indicates ICSF should audit usage events related to CKDS labels. An SMF type 82 subtype 44 record is logged for each event.
NO	No usage auditing of CKDS labels occurs.

`INTERVAL(n)`
Defines the time interval over which the audit records are aggregated. Specify *n* as a decimal value in hours from 1 through 24.

If the AUDITKEYUSGCKDS option is not specified, the default is AUDITKEYUSGCKDS(TOKEN(NO),LABEL(NO),INTERVAL(24)).

Note: An event that involves a token is considered to be any request that uses a token as opposed to a label. This is true regardless of Key Store Policy enablement.

For more information about the information contained in the audit record, see Appendix B in *z/OS Cryptographic Services ICSF System Programmer's Guide* for the description for the subtype 44 record.

The auditing of key usage events can also be controlled via the SETICSF operator command. See the description of the SETICSF command in *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

`AUDITKEYUSGPKDS(TOKEN(YES or NO),LABEL(YES or NO),INTERVAL(n))`

Provides a set of options that control auditing of events related to the usage of asymmetric CCA tokens. The audit logs are in the form of Type 82 SMF records.

`TOKEN(YES or NO)`

Controls usage auditing of PKDS tokens.

Value	Indication
YES	Indicates ICSF should audit usage events related to PKDS tokens. An SMF type 82 subtype 45 record is logged for each event.
NO	No usage auditing of PKDS tokens occurs.

`LABEL(YES or NO)`

Controls usage auditing of PKDS labels.

Value	Indication
YES	Indicates ICSF should audit usage events related to PKDS labels. An SMF type 82 subtype 45 record is logged for each event.
NO	No usage auditing of PKDS labels occurs.

`INTERVAL(n)`

Defines the time interval over which the audit records are aggregated. Specify *n* as a decimal value in hours from 1 through 24.

If the AUDITKEYUSGPKDS option is not specified, the default is AUDITKEYUSGPKDS(TOKEN(NO),LABEL(NO),INTERVAL(24)).

Note: An event that involves a token is considered to be any request that uses a token as opposed to a label. This is true regardless of Key Store Policy enablement.

For more information about the information contained in the audit record, see Appendix B in *z/OS Cryptographic Services ICSF System Programmer's Guide* for the description for the subtype 45 record.

The auditing of key usage events can also be controlled via the SETICSF operator command. See the description of the SETICSF command in *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

`AUDITPKCS11USG(TOKENOBJ(YES or NO),SESSIONOBJ(YES or NO),NOKEY(YES or NO),INTERVAL(n))`

Provides a set of options that control auditing of usage events related to PKCS #11 services. The audit logs are in the form of Type 82 SMF records.

`TOKEN(YES or NO)`

Controls usage auditing of PKCS #11 token objects.

Value	Indication
YES	Indicates ICSF should audit usage events related to PKCS #11 token objects. An SMF type 82 subtype 46 record is logged for each event.
NO	No usage auditing of PKCS #11 token objects occurs.

SESSIONOBJ(YES or NO)
Controls usage auditing of PKCS #11 session objects.

Value	Indication
YES	Indicates ICSF should audit usage events related to PKCS #11 session objects. An SMF type 82 subtype 46 record is logged for each event.
NO	No usage auditing of PKCS #11 session objects occurs.

NOKEY(YES or NO)
Controls usage auditing of PKCS #11 services that do not involve an object.

Value	Indication
YES	Indicates ICSF should audit relevant usages that do not pertain to a PKCS #11 object. Relevant usages include use of the PKCS #11 One-way hash, sign, or verify (CSFPPRF) and PKCS #11 Pseudo-random function (CSFPOWH) services. An SMF type 82 subtype 47 record is logged for each event.
NO	No usage auditing of PKCS #11 services that do not involve an object occurs.

INTERVAL(n)
Defines the time interval over which the audit records are aggregated. Specify *n* as a decimal value in hours from 1 through 24.

If the AUDITPKCS11USG option is not specified, the default is AUDITPKCS11USG(TOKENOBJ(NO),SESSIONOBJ(NO),NOKEY(NO), INTERVAL(24)).

For more information about the information contained in the audit record, see Appendix B in *z/OS Cryptographic Services ICSF System Programmer's Guide* for the description for the subtypes 46 and 47 records.

The auditing of key usage events can also be controlled via the SETICSF operator command. See the description of the SETICSF command in *z/OS Cryptographic Services ICSF System Programmer's Guide* for more information.

CHECKAUTH(YES or NO)
Indicates whether ICSF performs 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. If you specify CHECKAUTH(NO), the authorization checks against resources in the CSFSERV, the CSFKEYS, and the XCSFKEY classes 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).

COMPAT(YES, NO, or COEXIST)
Indicates whether ICSF is running in compatibility mode, noncompatibility

mode, or coexistence mode with the Programmed Cryptographic Facility (PCF). If you do not specify the COMPAT option, the default value is COMPAT(NO).

Value	**Indication**
YES	ICSF is running in compatibility mode, which means you can run CUSP and PCF applications on ICSF because ICSF supports the CUSP and PCF macros in this mode. You do not have to reassemble CUSP and PCF applications to do this. However, you cannot start CUSP or PCF at the same time as ICSF on the same MVS system.
NO	ICSF is running in noncompatibility mode, which means that 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. You can start PCF at the same time as ICSF on the same z/OS operating system. You can start ICSF and then start PCF or you can start PCF and then start CSF. You should use noncompatibility mode unless you are migrating from PCF to ICSF.
COEXIST	ICSF is running in coexistence mode. In this 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. In this mode, you can start PCF at the same time as ICSF on the same MVS system.

`CTRACE(CTICSFxx)`

Specifies the CTICSFxx ICSF CTRACE configuration data set to use from PARMLIB. CTICSF00 is the default ICSF CTRACE configuration data set that is installed with ICSF FMID HCR77A1 and later releases. CTICSF00 may be copied to create new PARMLIB members using the naming convention of CTICSFxx, where xx is a unique value specified by the user.

This parameter is optional. During ICSF startup, if this parameter is not specified, or if it is specified with a PARMLIB member that is absent or contains an incorrect option, ICSF CTRACE will attempt to use the default CTICSF00 PARMLIB member. If the CTICSF00 PARMLIB member is absent or contains an incorrect option, ICSF CTRACE will perform tracing using an internal default set of trace options. By default, ICSF CTRACE support will trace with the KdsIO, CardIO, and SysCall filters using a 2M buffer.

The operator console TRACE CT command may be used to dynamically change ICSF CTRACE options from a new PARMLIB member or directly from options specified on the command. If the TRACE CT command is used to specify a PARMLIB member that is either absent or contains incorrect options, ICSF CTRACE will ignore it and continue to use the current active options. If an incorrect option is specified directly with the TRACE CT command, ICSF CTRACE will ignore it as well and continue to use the current active options.

The Installation Options panel will display the current active PARMLIB member for CTRACE. If the TRACE CT command is used to update the CTRACE options, a value of “TRACE CT” will be displayed on the panel to indicate that the operator console TRACE CT command was used to modify the CTRACE options. Use the operator console DISPLAY TRACE,COMP=CSF command to display the current active CTRACE options.

Note: If the default CTICSF00 PARMLIB member has been deleted from the system and ICSF attempts to use it, ICSF CTRACE will perform tracing using an internal default set of trace options (KdsIO, CardIO, and SysCall filters using a 2M buffer). In this situation, if the operator console DISPLAY TRACE,COMP=CSF command is used to display the current active CTRACE options, a value of Minimum will be displayed.

`DEFAULTWRAP(internal_wrapping_method,external_wrapping_method)`

Specifies the default key wrapping for DES keys. Any token generated or updated by a service will be wrapped using the specified method unless overridden by rule array keyword or a skeleton token. The default wrapping method for internal and external tokens is specified independently.

Valid values for *internal_wrapping_method* and *external_wrapping_method* are:

ORIGINAL

Specifies the original CCA token wrapping be used: ECB wrapping for DES.

ENHANCED

Specifies the new X9.24 compliant CBC wrapping used. Note that the enhanced wrapping method requires an IBM zEnterprise 196 with a CEX3C.

`DOMAIN(n)`

Allows you to access one of the set of master key registers in the CCA and EP11 coprocessors. Each CCA domain contains AES, DES, ECC, and RSA master keys depending on the coprocessor licensed internal code level. Each EP11 domain contains the EP11 master key.

You can use domains to have separate master keys for different purposes.

You can use domains in basic mode or with PR/SM logical partition (LPAR) mode. In basic mode, you access only one domain at a time. You can specify a different master key in each domain. For example, you might have one master key for production operations and a different master key for test operations. In LPAR mode, you can have a different domain for each partition. The number you specify is the number of the domain to be used for this start of ICSF.

The DOMAIN parameter is an optional parameter in the installation options data set. 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. If you assign multiple domains to an LPAR, you can have separate master keys for different purposes.

You use the Crypto page of the Customize Activation Profile to assign a usage domain index to a logical partition and enable cryptographic functions. The DOMAIN number you specify in the installation options data set while running in a partition must be the same number as the usage domain index specified for the partition on the Crypto page.

To change and activate the other installation options, you must restart ICSF. In compatibility or coexistence mode, to change and activate the DOMAIN option, you must also re-IPL MVS. A re-IPL ensures that a program does not use a key that has been encrypted under a different master key to access a cryptographic service.

`FIPSMODE(YES or COMPAT or NO,FAIL(`*`fail-option`*`))`

Indicates whether z/OS PKCS #11 services must run in compliance with the Federal Information Processing Standard Security Requirements for Cryptographic Modules, referred to as FIPS 140-2. FIPS 140-2, published by the National Institute of Standards and Technology (NIST), is a standard that defines rules and restrictions for how cryptographic modules should protect sensitive or valuable information. The default is FIPSMODE(NO,FAIL(NO)).

By configuring z/OS PKCS #11 services to operate in compliance with FIPS 140-2 specifications, installations or individual applications can use the z/OS PKCS #11 services in a way that allows only the cryptographic algorithms (including key sizes) approved by the standard, and restricts access to the algorithms that are not approved. For more information, refer to *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

KDSREFDAYS

Specifies, in days, how often a record should be written for a reference date/time change. A key is referenced when it is used to perform a cryptographic operation or read, such that the retrieved token may have been used in a cryptographic operation. If a key is referenced ICSF will check the date and time the key was referenced previous to the current reference. If the number of days between the current date and time and the date and time the key was last referenced is greater than or equal to the number of days specified in the KDSREFDAYS installation option then the key reference date/time in the KDS will be updated to the current date and time. Otherwise the reference date/time will remain the same. Note, in this context days are 24 hour periods not necessarily beginning or ending at midnight.

Example: If KDSREFDAYS(7) was specified and a key was referenced on Monday, January 1st at 8 AM, and the reference date/time for the key was updated at that time, then any key reference before Monday, January 8th at 8 AM (7 days) will not update the reference date/time in the key record. If the key is referenced again at 7:50 AM on Monday, January 8th, the reference date/time for the key in the KDS will remain January 1st at 8 AM because fewer than seven days have passed. The reference date/time will not be updated until the next time the key is used again Monday, January 8th at 8 AM or after.

KDSREFDAYS applies to all KDS that are in the format that supports key reference tracking. In an environment of mixed KDS formats, where some support reference date tracking and some do not (for example, the CKDS supports reference date tracking, but the PKDS does not) key references will not be tracked for keys in a KDS does not support it, regardless on the value of KDSREFDAYS, until that KDS is updated to the new format. In a SYSPLEX, all systems must be started with the same value of KDSREFDAYS to ensure proper tracking of reference date/times.

KDSREFDAYS(0) means that ICSF will not keep track of key reference dates. The default is KDSREFDAYS(1). The maximum value allowed is KDSREFDAYS(30).

Note: Updates to records using the Key Generator Utility Program (KGUP) are not subject to the value specified in the KDSREFDAYS option. All updates made via KGUP will update the reference date/time if the CKDS is in a format that supports reference date tracking (KDSR).

KEYARCHMSG(YES or NO)

Controls whether a joblog message is issued when an application successfully references a key data set record that has been archived. The message is only issued for the first successful reference of a record. The results of the service request is not affected by this control. The default is NO.

Value	Indication
YES	ICSF issues a message the first time an archived record is referenced by an application.

NO	ICSF does not issue a message when an archived record is referenced by an application.

MASTERKCVLEN(n)
Specifies the length of the master key verification patterns.

Note: If your system is running ICSF FMID HCR77B1 or later, the DISPLAY ICSF,OPT operator command can also be used to display the value of this option. The SETICSF OPT,MKCVLEN operation command can be used to set the value of this option. For additional information on ICSF operator commands, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

MAXSESSOBJECTS(n)
Defines the maximum number of PKCS #11 session objects and states an unauthorized (problem state, non-system key) application may own at any one time. Specify `n` as a decimal value from 1024 through 2147483647. If you do not specify the MAXSESSOBJECTS option, the default value is MAXSESSOBJECTS(65535).

REASONCODES(ICSF or TSS)

Specifies which set of reason codes the application interface returns.

Value	Indication
ICSF	ICSF reason codes are returned.
TSS	Reason codes used by the IBM 4765 PCIe, IBM 4767 PCIe, and IBM 4764 PCI-X Cryptographic Coprocessors are returned.

ICSF is the default.

RNGCACHE(YES or NO)
Indicates whether ICSF should maintain a cache of random numbers to be used by services that require them. When YES is specified for this option, a noticeable performance improvement may be realized by workloads requesting a significant amount of random data.

If you do not specify the RNGCACHE option, the default value is RNGCACHE(YES).

Value	Indication
YES	ICSF maintains a random number cache.
NO	ICSF does not maintain a random number cache.

SSM(YES or NO)
Indicates whether special secure mode is enabled. This mode lowers the security of your system. It allows you to input clear keys by using KGUP, produce clear PINs, and use the Secure Key Import callable services. If you do not specify the SSM option, the default value is SSM(NO).

Value	Indication
YES	Special secure mode is enabled. SSM(YES) must be specified in order to use KGUP, Secure Key Import callable services, and Clear PIN Generate callable service.
NO	Special secure mode is disabled. The SSM option can be changed from NO to YES while ICSF is running by defining the CSF.SSM.ENABLE SAF profile within the

XFACILIT resource class. To revert to your startup option, delete the CSF.SSM.ENABLE profile. The XFACILIT class must be refreshed after each change for it to take effect.

Note: When using the SAF profiles to set the SSM, all ICSF instances sharing the SAF profile will be affected.

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

Displays the current value of the SYSPLEXCKDS option. The values of the option can be YES or NO, with the default being NO. If SYSPLEXCKDS(NO,FAIL(*fail-option*)) is specified, no XCF signalling will be performed when an update to a CKDS record occurs. If SYSPLEXCKDS(YES,FAIL(*fail-option*)) is specified, the support described in "CKDS management in a sysplex" on page 174 will occur.
The fail-option can be specified as either YES or NO. If FAIL(YES) is specified then ICSF initialization will end abnormally if the request during ICSF initialization to join the ICSF sysplex group fails. If FAIL(NO) is specified, then ICSF initialization processing will continue even if the request to join the ICSF sysplex group fails. This system will not be notified of updates to the CKDS by other members of the ICSF sysplex group. The default is SYSPLEXCKDS(NO,FAIL(NO)).

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

Displays the current value of the SYSPLEXPKDS option. The values of the option can be YES or NO, with the default being NO. If SYSPLEXPKDS(NO,FAIL(*fail-option*)) is specified, no XCF signalling will be performed when an update to a PKDS record occurs. If SYSPLEXPKDS(YES,FAIL(*fail-option*)) is specified, the support described in "PKDS management in a sysplex" on page 178 will occur.
The fail-option can be specified as either YES or NO. If FAIL(YES) is specified then ICSF initialization will end abnormally if the request during ICSF initialization to join the ICSF sysplex group fails. If FAIL(NO) is specified, then ICSF initialization processing will continue even if the request to join the ICSF sysplex group fails. This system will not be notified of updates to the PKDS by other members of the ICSF sysplex group. The default is SYSPLEXPKDS(NO,FAIL(NO)).

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

Displays the current value of the SYSPLEXTKDS option. 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, the support described in "TKDS management in a sysplex" on page 181 will occur.
The fail-option can be specified as either YES or NO. If FAIL(YES) is specified then ICSF initialization will end abnormally if the request during ICSF initialization to join the ICSF sysplex group fails. If FAIL(NO) is specified, then ICSF initialization processing will continue even if the request to join the ICSF sysplex group fails. This system will not be notified of updates to the TKDS by other members of the ICSF sysplex group. The default is SYSPLEXTKDS(NO,FAIL(NO)).

USERPARM(value)

Displays the value of an 8-byte field that is defined for installation use. ICSF stores this value in the CCVT_USERPARM field of the Cryptographic

Communication Vector Table (CCVT). An application program or installation exit can examine this field and use it to set system environment information.

WAITLIST(value)

Displays the current value of the WAITLIST option. If WAITLIST is coded, the value will be “dataset” and a second line will contain the name of the specified Wait List data set. If WAITLIST is not coded, the value will be “default”. If the data set specified by the WAITLIST option cannot be allocated or opened, the value will also be “default”.

For more information about the ICSF startup procedure and installation options, see *z/OS Cryptographic Services ICSF System Programmer's Guide*. At any time while you are running ICSF, you can check the current value of these installation options.

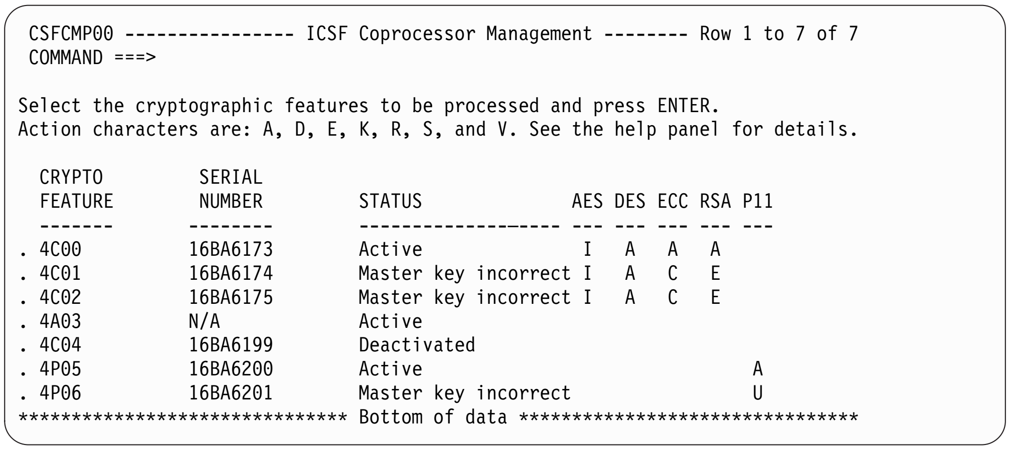
The installation exits and installation-defined callable services are also specified in the installation options data set, but they are not displayed on this panel. For a description of how to display the installation exit information, see “Displaying installation exits” on page 302. For a description of how to display installation-defined callable service information, see “Displaying installation-defined callable services” on page 306.

Note: If your system is running ICSF FMID HCR77B1 or later, the DISPLAY ICSF,OPTIONS operator command can also be used to display a subset of ICSF options. For additional information on ICSF operator commands, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Display CCA domain roles

Use the ICSF panels to display the coprocessor role for the coprocessor. All the access control points enabled will be listed.

1. Select option 1, COPROCESSOR MGMT, on the “ICSF Primary Menu panel” on page 387.
2. The Coprocessor Management panel appears. Refer to Figure 99.



```
 CSFCMP00 ---------------- ICSF Coprocessor Management -------- Row 1 to 7 of 7
 COMMAND ===>

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

  CRYPTO         SERIAL
  FEATURE        NUMBER        STATUS             AES DES ECC RSA P11
  -------       --------       ------------------- --- --- --- --- ---
. 4C00          16BA6173       Active               I   A   A   A
. 4C01          16BA6174       Master key incorrect I   A   C   E
. 4C02          16BA6175       Master key incorrect I   A   C   E
. 4A03          N/A            Active
. 4C04          16BA6199       Deactivated
. 4P05          16BA6200       Active                               A
. 4P06          16BA6201       Master key incorrect                 U
******************************* Bottom of data ********************************
```

Figure 99. Coprocessor Management Panel

3. Select the desired coprocessor by entering an 'R' or 'V' to the left of the coprocessor.

The display shown when 'R' is used (see Figure 100) lists all of the enabled access controls in alphabetic order.

The display shown when 'V' is used (see Figure 104 on page 296) lists all of the enabled access controls and the offset within the role.The list can be ordered by the access control name or the offset. Press enter and the Domain Role Display panel appears.

Note: A TKE workstation is required in order to change the coprocessor role. See *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

```
Access Control Manager - Read role
Authorize UDX
AES Master Key - Clear new master key register
AES Master Key - Combine key parts
AES Master Key - Load first key part
AES Master Key - Set master key
Clear Key Import/Multiple Clear Key Import - DES
Clear PIN Encrypt
Clear PIN Generate - GBP
Clear PIN Generate - Interbank
Clear PIN Generate - VISA PVV
Clear PIN Generate - 3624
Clear PIN Generate Alternate - VISA PVV
Clear PIN Generate Alternate - 3624 Offset
Control Vector Translate
Cryptographic Variable Encipher
CKDS Conversion2 - Allow use of REFORMAT
CKDS Conversion2 - Allow wrapping override keywords
CKDS Conversion2 - Convert from enhanced to original
CVV Key Combine
CVV Key Combine - Allow wrapping override keywords
CVV Key Combine - Permit mixed key types
Data Key Export
Data Key Export - Unrestricted
Data Key Import
Data Key Import - Unrestricted
Decipher - DES
Digital Signature Generate
Digital Signature Verify
Diversified Key Generate - Allow wrapping override keywords
Diversified Key Generate - CLR8-ENC
Diversified Key Generate - Single length or same halves
Diversified Key Generate - SESS-XOR
Diversified Key Generate - TDES-DEC
Diversified Key Generate - TDES-ENC
Diversified Key Generate - TDES-XOR
Diversified Key Generate - TDESEMV2/TDESEMV4
DATAM Key Management Control
DES Master Key - Clear new master key register
DES Master Key - Combine key parts
DES Master Key - Load first key part
DES Master Key - Set master key
DUKPT - PIN Verify, PIN Translate
Encipher - DES
Encrypted PIN Generate - GBP
Encrypted PIN Generate - Interbank
Encrypted PIN Generate - 3624
Encrypted PIN Translate - Reformat
```

Figure 100. CCA Coprocessor Role Display panel

```
Encrypted PIN Translate - Translate
Encrypted PIN Verify - GBP
Encrypted PIN Verify - Interbank
Encrypted PIN Verify - VISA PVV
Encrypted PIN Verify - 3624
ECC Diffie-Hellman
ECC Diffie-Hellman - Allow key wrap override
ECC Diffie-Hellman - Allow BP Curve 160
ECC Diffie-Hellman - Allow BP Curve 192
ECC Diffie-Hellman - Allow BP Curve 224
ECC Diffie-Hellman - Allow BP Curve 256
ECC Diffie-Hellman - Allow BP Curve 320
ECC Diffie-Hellman - Allow BP Curve 384
ECC Diffie-Hellman - Allow BP Curve 512
ECC Diffie-Hellman - Allow Prime Curve 192
ECC Diffie-Hellman - Allow Prime Curve 224
ECC Diffie-Hellman - Allow Prime Curve 256
ECC Diffie-Hellman - Allow Prime Curve 384
ECC Diffie-Hellman - Allow Prime Curve 521
ECC Diffie-Hellman - Allow PASSTHRU
ECC Master Key - Clear new master key register
ECC Master Key - Combine key parts
ECC Master Key - Load first key part
ECC Master Key - Set master key
HMAC Generate - SHA-1
HMAC Generate - SHA-224
HMAC Generate - SHA-256
HMAC Generate - SHA-384
HMAC Generate - SHA-512
HMAC Verify - SHA-1
HMAC Verify - SHA-224
HMAC Verify - SHA-256
HMAC Verify - SHA-384
HMAC Verify - SHA-512
Key Export
Key Export - Unrestricted
Key Generate - Key set
Key Generate - Key set extended
Key Generate - OP
Key Generate - SINGLE-R
Key Generate2 - Key set
Key Generate2 - OP
Key Import
Key Import - Unrestricted
Key Part Import - first key part
Key Part Import - middle and last
Key Part Import - Allow wrapping override keywords
Key Part Import - ADD-PART
Key Part Import - COMPLETE
Key Part Import - Unrestricted
Key Part Import2 - Add last required key part
Key Part Import2 - Add optional key part
Key Part Import2 - Add second of 3 or more key parts
Key Part Import2 - Complete key
Key Part Import2 - Load first key part, require 1 key parts
Key Part Import2 - Load first key part, require 2 key parts
Key Part Import2 - Load first key part, require 3 key parts
Key Test and Key Test2
Key Test2 - AES, ENC-ZERO
Key Translate
Key Translate2
Key Translate2 - Allow use of REFORMAT
Key Translate2 - Allow wrapping override keywords
Multiple Clear Key Import - Allow wrapping override keywords
Multiple Clear Key Import/Multiple Secure Key Import - AES
Multiple Secure Key Import - Allow wrapping override keywords
MAC Generate
MAC Verify
```

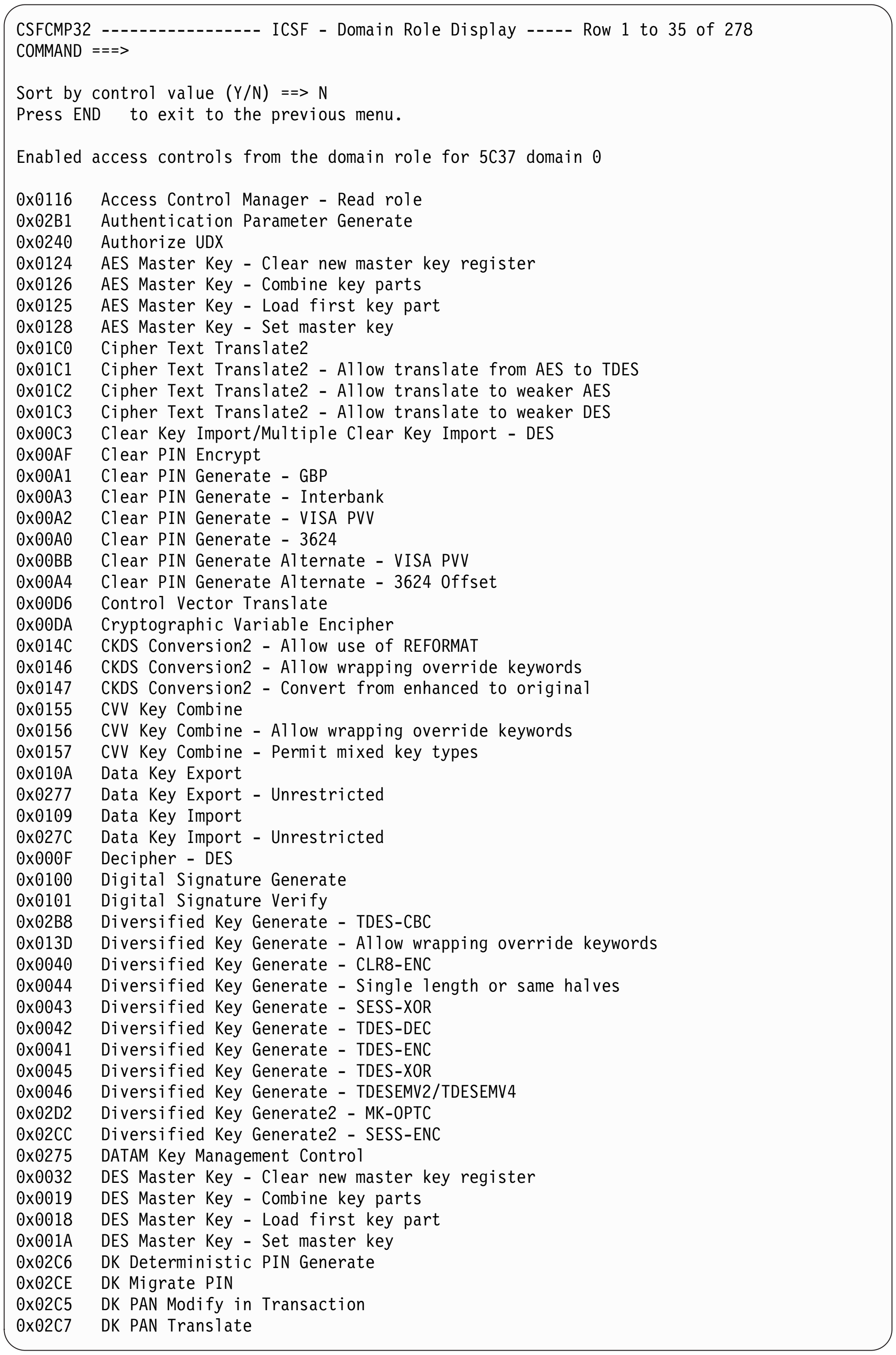
Figure 101. CCA Coprocessor Role Display panel - part 2

```
NOCV KEK usage for export-related functions
NOCV KEK usage for import-related functions
Operational Key Load
Prohibit Export
Prohibit Export Extended
PCF CKDS conversion utility
PCF CKDS Conversion - Allow wrapping override keywords
PIN Change/Unblock - change EMV PIN with IPINENC
PIN Change/Unblock - change EMV PIN with OPINENC
PKA Decrypt
PKA Encrypt
PKA Key Generate
PKA Key Generate - Clear ECC keys
PKA Key Generate - Clear RSA keys
PKA Key Generate - Clone
PKA Key Generate - Permit Regeneration Data
PKA Key Generate - Permit Regeneration Data Retain
PKA Key Import
PKA Key Import - Import an external trusted block
PKA Key Token Change RTCMK
PKA Key Translate - from source EXP KEK to target EXP KEK
PKA Key Translate - from source IMP KEK to target EXP KEK
PKA Key Translate - from source IMP KEK to target IMP KEK
PKA Key Translate - from CCA RSA to SC CRT Format
PKA Key Translate - from CCA RSA to SC ME Format
PKA Key Translate - from CCA RSA to SC Visa Format
Reencipher CKDS2
Reencipher CKDS
Reencipher PKDS
Remote Key Export - Gen or export a non-CCA node key
Restrict Key Attribute - Export Control
Restrict Key Attribute - Permit setting the TR-31 export bit
Retained Key Delete
Retained Key List
RSA Master Key - Clear new master key register
RSA Master Key - Combine key parts
RSA Master Key - Load first key part
RSA Master Key - Set master key
Secure Key Import - DES,IM
Secure Key Import - DES,OP
Secure Key Import2 - IM
Secure Key Import2 - OP
Secure Messaging for Keys
Secure Messaging for PINs
Symmetric token wrapping - external enhanced method
Symmetric token wrapping - external original method
Symmetric token wrapping - internal enhanced method
Symmetric token wrapping - internal original method
Symmetric Algorithm Decipher - secure AES keys
Symmetric Algorithm Encipher - secure AES keys
Symmetric Key Encipher/Decipher - Encrypted AES keys
Symmetric Key Encipher/Decipher - Encrypted DES keys
Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2
Symmetric Key Export - AES, ZERO-PAD
Symmetric Key Export - AES,PKOAEP2
Symmetric Key Export - AESKW
Symmetric Key Export - DES, PKCS-1.2
Symmetric Key Export - DES, ZERO-PAD
Symmetric Key Export - HMAC,PKOAEP2
```

Figure 102. CCA Coprocessor Role Display panel – part 3

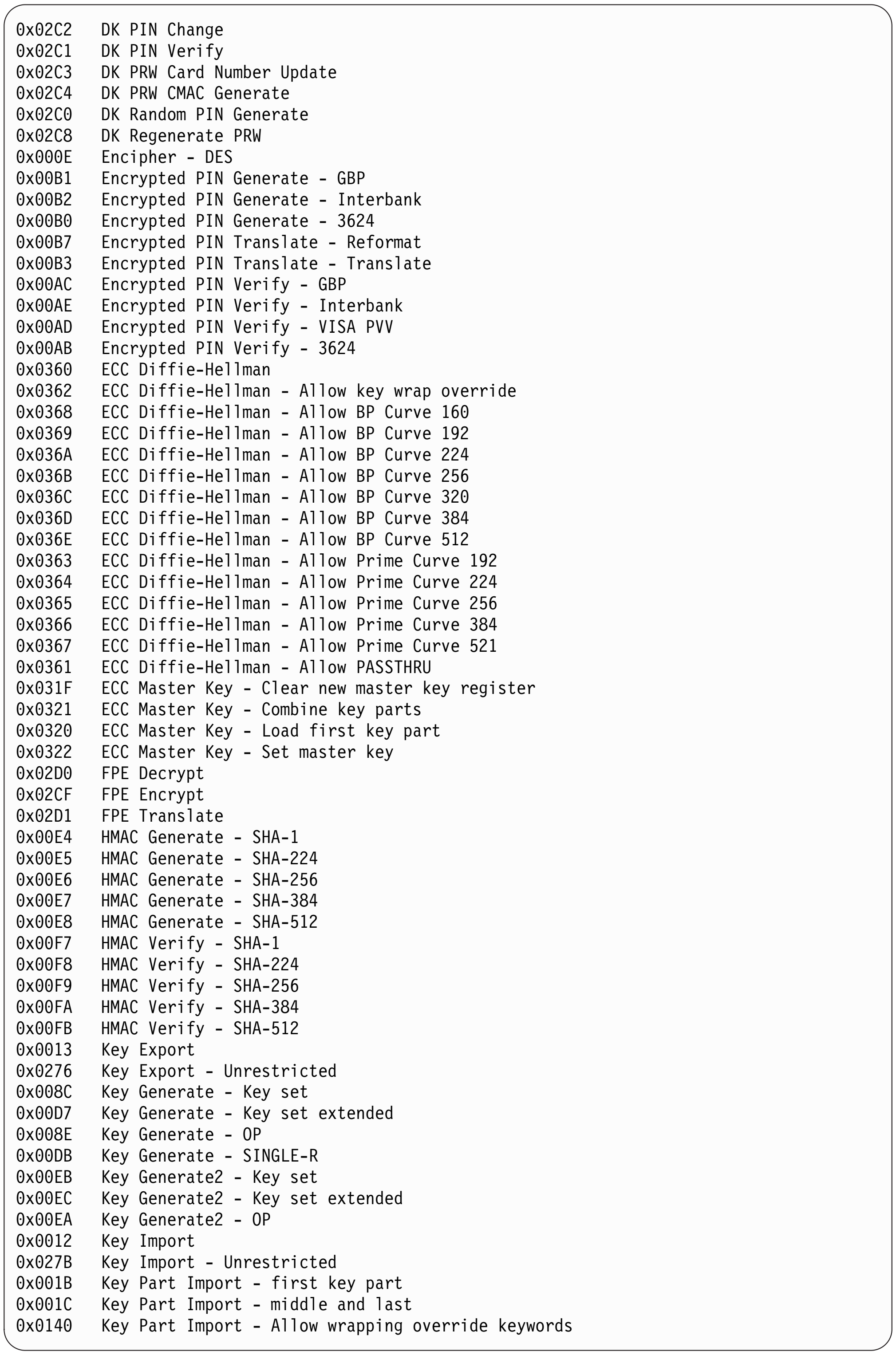
```
Symmetric Key Generate - Allow wrapping override keywords
Symmetric Key Generate - AES, PKCSOAEP, PKCS-1.2
Symmetric Key Generate - AES, ZERO-PAD
Symmetric Key Generate - DES, PKA92
Symmetric Key Generate - DES, PKCS-1.2
Symmetric Key Generate - DES, ZERO-PAD
Symmetric Key Import - Allow wrapping override keywords
Symmetric Key Import - AES, PKCSOAEP, PKCS-1.2
Symmetric Key Import - AES, ZERO-PAD
Symmetric Key Import - DES, PKA92 KEK
Symmetric Key Import - DES, PKCS-1.2
Symmetric Key Import - DES, ZERO-PAD
Symmetric Key Import2 - AES,PKOAEP2
Symmetric Key Import2 - AESKW
Symmetric Key Import2 - HMAC,PKOAEP2
Symmetric Key Token Change - RTCMK
Symmetric Key Token Change2 - RTCMK
SET Block Compose
SET Block Decompose
SET Block Decompose - PIN Extension IPINENC
SET Block Decompose - PIN Extension OPINENC
Transaction Validation - Generate
Transaction Validation - Verify CSC-3
Transaction Validation - Verify CSC-4
Transaction Validation - Verify CSC-5
Trusted Block Create - Activate an inactive block
Trusted Block Create - Create Block in inactive form
TR31 Export - Permit any CCA key if INCL-CV is specified
TR31 Export - Permit version A TR-31 key blocks
TR31 Export - Permit version B TR-31 key blocks
TR31 Export - Permit version C TR-31 key blocks
TR31 Export - Permit DATA to C0:G/C
TR31 Export - Permit DATA to D0:B
TR31 Export - Permit DKYGENKY:DKYL0+DALL to E4
TR31 Export - Permit DKYGENKY:DKYL0+DALL to E5
TR31 Export - Permit DKYGENKY:DKYL0+DDATA to E4
TR31 Export - Permit ENCIPHER/DECIPHER/CIPHER to D0:E/D/B
TR31 Export - Permit IPINENC to P0:D
TR31 Export - Permit KEYGENKY:DUKPT to B0
TR31 Export - Permit MAC/DATA/DATAM to M1:G/C
TR31 Export - Permit MAC/DATA/DATAM to M3:G/C
TR31 Export - Permit MAC/MACVER:ANY-MAC to C0:G/C/V
TR31 Export - Permit MACVER/DATAMV to M0:V
TR31 Export - Permit MACVER/DATAMV to M1:V
TR31 Export - Permit MACVER/DATAMV to M3:V
TR31 Export - Permit OPINENC to P0:E
TR31 Export - Permit PINGEN:NO-SPEC/IBM-PIN/IBM-PINO to V1
TR31 Export - Permit PINGEN:NO-SPEC/VISA-PVV to V2
TR31 Export - Permit PINVER:NO-SPEC/IBM-PIN/IBM-PINO to V1
TR31 Export - Permit PINVER:NO-SPEC/VISA-PVV to V2
TR31 Import - Permit override of default wrapping method
TR31 Import - Permit version A TR-31 key blocks
TR31 Import - Permit version B TR-31 key blocks
TR31 Import - Permit version C TR-31 key blocks
TR31 Import - Permit E4 to DKYGENKY:DKYL0+DDATA
TR31 Import - Permit M0/M1/M3 to MAC/MACVER:ANY-MAC
TR31 Import - Permit P0:D to IPINENC
TR31 Import - Permit P0:E to OPINENC
TR31 Import - Permit V1 to PINGEN:IBM-PIN/IBM-PINO
TR31 Import - Permit V1 to PINVER:IBM-PIN/IBM-PINO
TR31 Import - Permit V2 to PINGEN:VISA-PVV
TR31 Import - Permit V2 to PINVER:VISA-PVV
VISA CVV Generate
VISA CVV Verify
```

Figure 103. CCA Coprocessor Role Display panel – part 4



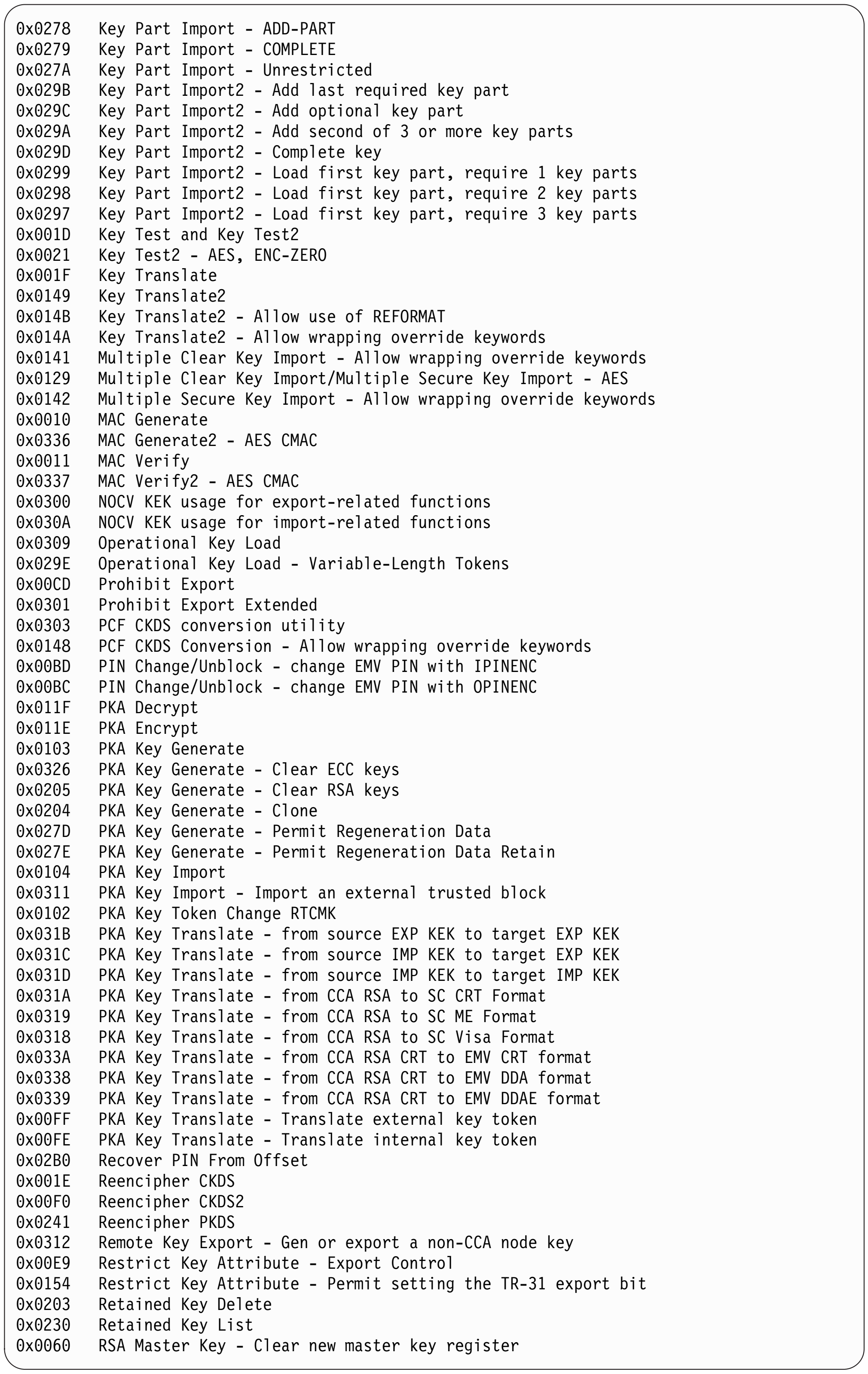
```
CSFCMP32 ----------------- ICSF - Domain Role Display ----- Row 1 to 35 of 278
COMMAND ===>

Sort by control value (Y/N) ==> N
Press END   to exit to the previous menu.

Enabled access controls from the domain role for 5C37 domain 0

0x0116   Access Control Manager - Read role
0x02B1   Authentication Parameter Generate
0x0240   Authorize UDX
0x0124   AES Master Key - Clear new master key register
0x0126   AES Master Key - Combine key parts
0x0125   AES Master Key - Load first key part
0x0128   AES Master Key - Set master key
0x01C0   Cipher Text Translate2
0x01C1   Cipher Text Translate2 - Allow translate from AES to TDES
0x01C2   Cipher Text Translate2 - Allow translate to weaker AES
0x01C3   Cipher Text Translate2 - Allow translate to weaker DES
0x00C3   Clear Key Import/Multiple Clear Key Import - DES
0x00AF   Clear PIN Encrypt
0x00A1   Clear PIN Generate - GBP
0x00A3   Clear PIN Generate - Interbank
0x00A2   Clear PIN Generate - VISA PVV
0x00A0   Clear PIN Generate - 3624
0x00BB   Clear PIN Generate Alternate - VISA PVV
0x00A4   Clear PIN Generate Alternate - 3624 Offset
0x00D6   Control Vector Translate
0x00DA   Cryptographic Variable Encipher
0x014C   CKDS Conversion2 - Allow use of REFORMAT
0x0146   CKDS Conversion2 - Allow wrapping override keywords
0x0147   CKDS Conversion2 - Convert from enhanced to original
0x0155   CVV Key Combine
0x0156   CVV Key Combine - Allow wrapping override keywords
0x0157   CVV Key Combine - Permit mixed key types
0x010A   Data Key Export
0x0277   Data Key Export - Unrestricted
0x0109   Data Key Import
0x027C   Data Key Import - Unrestricted
0x000F   Decipher - DES
0x0100   Digital Signature Generate
0x0101   Digital Signature Verify
0x02B8   Diversified Key Generate - TDES-CBC
0x013D   Diversified Key Generate - Allow wrapping override keywords
0x0040   Diversified Key Generate - CLR8-ENC
0x0044   Diversified Key Generate - Single length or same halves
0x0043   Diversified Key Generate - SESS-XOR
0x0042   Diversified Key Generate - TDES-DEC
0x0041   Diversified Key Generate - TDES-ENC
0x0045   Diversified Key Generate - TDES-XOR
0x0046   Diversified Key Generate - TDESEMV2/TDESEMV4
0x02D2   Diversified Key Generate2 - MK-OPTC
0x02CC   Diversified Key Generate2 - SESS-ENC
0x0275   DATAM Key Management Control
0x0032   DES Master Key - Clear new master key register
0x0019   DES Master Key - Combine key parts
0x0018   DES Master Key - Load first key part
0x001A   DES Master Key - Set master key
0x02C6   DK Deterministic PIN Generate
0x02CE   DK Migrate PIN
0x02C5   DK PAN Modify in Transaction
0x02C7   DK PAN Translate
```

Figure 104. CCA Domain Role Display panel



```
0x02C2   DK PIN Change
0x02C1   DK PIN Verify
0x02C3   DK PRW Card Number Update
0x02C4   DK PRW CMAC Generate
0x02C0   DK Random PIN Generate
0x02C8   DK Regenerate PRW
0x000E   Encipher - DES
0x00B1   Encrypted PIN Generate - GBP
0x00B2   Encrypted PIN Generate - Interbank
0x00B0   Encrypted PIN Generate - 3624
0x00B7   Encrypted PIN Translate - Reformat
0x00B3   Encrypted PIN Translate - Translate
0x00AC   Encrypted PIN Verify - GBP
0x00AE   Encrypted PIN Verify - Interbank
0x00AD   Encrypted PIN Verify - VISA PVV
0x00AB   Encrypted PIN Verify - 3624
0x0360   ECC Diffie-Hellman
0x0362   ECC Diffie-Hellman - Allow key wrap override
0x0368   ECC Diffie-Hellman - Allow BP Curve 160
0x0369   ECC Diffie-Hellman - Allow BP Curve 192
0x036A   ECC Diffie-Hellman - Allow BP Curve 224
0x036B   ECC Diffie-Hellman - Allow BP Curve 256
0x036C   ECC Diffie-Hellman - Allow BP Curve 320
0x036D   ECC Diffie-Hellman - Allow BP Curve 384
0x036E   ECC Diffie-Hellman - Allow BP Curve 512
0x0363   ECC Diffie-Hellman - Allow Prime Curve 192
0x0364   ECC Diffie-Hellman - Allow Prime Curve 224
0x0365   ECC Diffie-Hellman - Allow Prime Curve 256
0x0366   ECC Diffie-Hellman - Allow Prime Curve 384
0x0367   ECC Diffie-Hellman - Allow Prime Curve 521
0x0361   ECC Diffie-Hellman - Allow PASSTHRU
0x031F   ECC Master Key - Clear new master key register
0x0321   ECC Master Key - Combine key parts
0x0320   ECC Master Key - Load first key part
0x0322   ECC Master Key - Set master key
0x02D0   FPE Decrypt
0x02CF   FPE Encrypt
0x02D1   FPE Translate
0x00E4   HMAC Generate - SHA-1
0x00E5   HMAC Generate - SHA-224
0x00E6   HMAC Generate - SHA-256
0x00E7   HMAC Generate - SHA-384
0x00E8   HMAC Generate - SHA-512
0x00F7   HMAC Verify - SHA-1
0x00F8   HMAC Verify - SHA-224
0x00F9   HMAC Verify - SHA-256
0x00FA   HMAC Verify - SHA-384
0x00FB   HMAC Verify - SHA-512
0x0013   Key Export
0x0276   Key Export - Unrestricted
0x008C   Key Generate - Key set
0x00D7   Key Generate - Key set extended
0x008E   Key Generate - OP
0x00DB   Key Generate - SINGLE-R
0x00EB   Key Generate2 - Key set
0x00EC   Key Generate2 - Key set extended
0x00EA   Key Generate2 - OP
0x0012   Key Import
0x027B   Key Import - Unrestricted
0x001B   Key Part Import - first key part
0x001C   Key Part Import - middle and last
0x0140   Key Part Import - Allow wrapping override keywords
```

Figure 105. CCA Domain Role Display panel - part 2

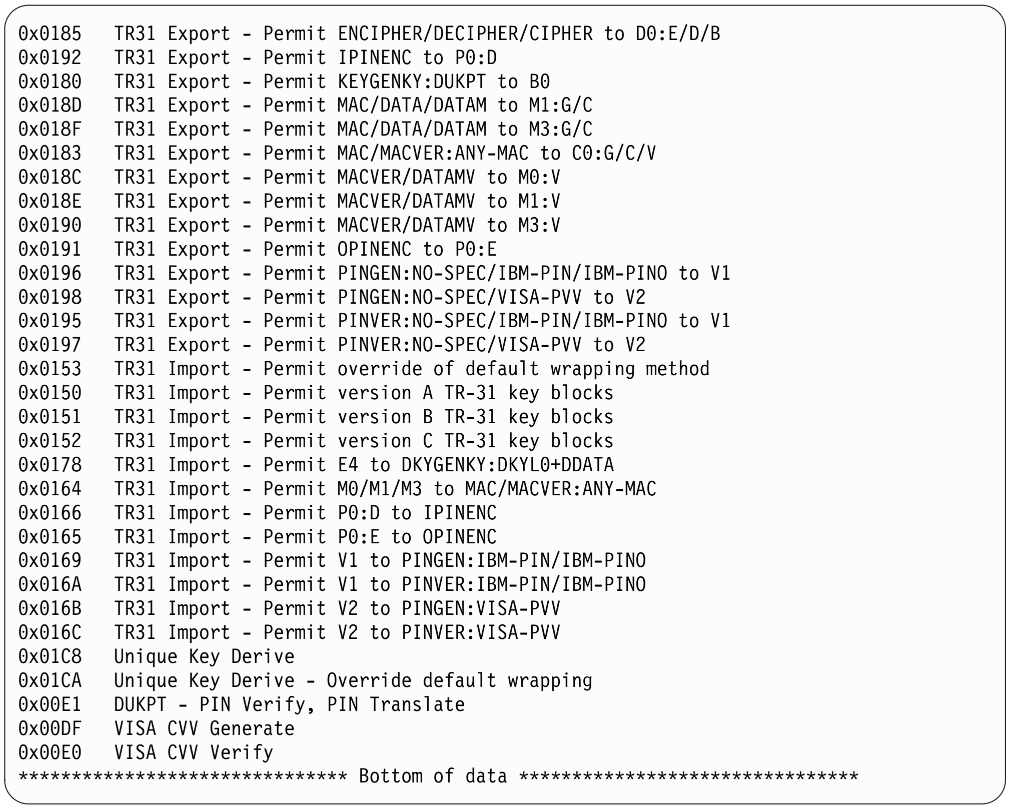


```
0x0278   Key Part Import - ADD-PART
0x0279   Key Part Import - COMPLETE
0x027A   Key Part Import - Unrestricted
0x029B   Key Part Import2 - Add last required key part
0x029C   Key Part Import2 - Add optional key part
0x029A   Key Part Import2 - Add second of 3 or more key parts
0x029D   Key Part Import2 - Complete key
0x0299   Key Part Import2 - Load first key part, require 1 key parts
0x0298   Key Part Import2 - Load first key part, require 2 key parts
0x0297   Key Part Import2 - Load first key part, require 3 key parts
0x001D   Key Test and Key Test2
0x0021   Key Test2 - AES, ENC-ZERO
0x001F   Key Translate
0x0149   Key Translate2
0x014B   Key Translate2 - Allow use of REFORMAT
0x014A   Key Translate2 - Allow wrapping override keywords
0x0141   Multiple Clear Key Import - Allow wrapping override keywords
0x0129   Multiple Clear Key Import/Multiple Secure Key Import - AES
0x0142   Multiple Secure Key Import - Allow wrapping override keywords
0x0010   MAC Generate
0x0336   MAC Generate2 - AES CMAC
0x0011   MAC Verify
0x0337   MAC Verify2 - AES CMAC
0x0300   NOCV KEK usage for export-related functions
0x030A   NOCV KEK usage for import-related functions
0x0309   Operational Key Load
0x029E   Operational Key Load - Variable-Length Tokens
0x00CD   Prohibit Export
0x0301   Prohibit Export Extended
0x0303   PCF CKDS conversion utility
0x0148   PCF CKDS Conversion - Allow wrapping override keywords
0x00BD   PIN Change/Unblock - change EMV PIN with IPINENC
0x00BC   PIN Change/Unblock - change EMV PIN with OPINENC
0x011F   PKA Decrypt
0x011E   PKA Encrypt
0x0103   PKA Key Generate
0x0326   PKA Key Generate - Clear ECC keys
0x0205   PKA Key Generate - Clear RSA keys
0x0204   PKA Key Generate - Clone
0x027D   PKA Key Generate - Permit Regeneration Data
0x027E   PKA Key Generate - Permit Regeneration Data Retain
0x0104   PKA Key Import
0x0311   PKA Key Import - Import an external trusted block
0x0102   PKA Key Token Change RTCMK
0x031B   PKA Key Translate - from source EXP KEK to target EXP KEK
0x031C   PKA Key Translate - from source IMP KEK to target EXP KEK
0x031D   PKA Key Translate - from source IMP KEK to target IMP KEK
0x031A   PKA Key Translate - from CCA RSA to SC CRT Format
0x0319   PKA Key Translate - from CCA RSA to SC ME Format
0x0318   PKA Key Translate - from CCA RSA to SC Visa Format
0x033A   PKA Key Translate - from CCA RSA CRT to EMV CRT format
0x0338   PKA Key Translate - from CCA RSA CRT to EMV DDA format
0x0339   PKA Key Translate - from CCA RSA CRT to EMV DDAE format
0x00FF   PKA Key Translate - Translate external key token
0x00FE   PKA Key Translate - Translate internal key token
0x02B0   Recover PIN From Offset
0x001E   Reencipher CKDS
0x00F0   Reencipher CKDS2
0x0241   Reencipher PKDS
0x0312   Remote Key Export - Gen or export a non-CCA node key
0x00E9   Restrict Key Attribute - Export Control
0x0154   Restrict Key Attribute - Permit setting the TR-31 export bit
0x0203   Retained Key Delete
0x0230   Retained Key List
0x0060   RSA Master Key - Clear new master key register
```

Figure 106. CCA Domain Role Display panel - part 3

```
0x0054   RSA Master Key - Combine key parts
0x0053   RSA Master Key - Load first key part
0x0057   RSA Master Key - Set master key
0x00DC   Secure Key Import - DES,IM
0x00C4   Secure Key Import - DES,OP
0x00F3   Secure Key Import2 - IM
0x00F2   Secure Key Import2 - OP
0x0273   Secure Messaging for Keys
0x0274   Secure Messaging for PINs
0x013B   Symmetric token wrapping - external enhanced method
0x013C   Symmetric token wrapping - external original method
0x0139   Symmetric token wrapping - internal enhanced method
0x013A   Symmetric token wrapping - internal original method
0x012B   Symmetric Algorithm Decipher - secure AES keys
0x012A   Symmetric Algorithm Encipher - secure AES keys
0x0296   Symmetric Key Encipher/Decipher - Encrypted AES keys
0x0295   Symmetric Key Encipher/Decipher - Encrypted DES keys
0x0130   Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2
0x0131   Symmetric Key Export - AES, ZERO-PAD
0x00FC   Symmetric Key Export - AES,PKOAEP2
0x0327   Symmetric Key Export - AESKW
0x02B3   Symmetric Key Export - AESKWCV
0x0105   Symmetric Key Export - DES, PKCS-1.2
0x023E   Symmetric Key Export - DES, ZERO-PAD
0x00F5   Symmetric Key Export - HMAC,PKOAEP2
0x02B5   Symmetric Key Export with Data
0x02B6   Symmetric Key Export with Data - Special
0x013E   Symmetric Key Generate - Allow wrapping override keywords
0x012C   Symmetric Key Generate - AES, PKCSOAEP, PKCS-1.2
0x012D   Symmetric Key Generate - AES, ZERO-PAD
0x010D   Symmetric Key Generate - DES, PKA92
0x023F   Symmetric Key Generate - DES, PKCS-1.2
0x023C   Symmetric Key Generate - DES, ZERO-PAD
0x0144   Symmetric Key Import - Allow wrapping override keywords
0x012E   Symmetric Key Import - AES, PKCSOAEP, PKCS-1.2
0x012F   Symmetric Key Import - AES, ZERO-PAD
0x0235   Symmetric Key Import - DES, PKA92 KEK
0x0106   Symmetric Key Import - DES, PKCS-1.2
0x023D   Symmetric Key Import - DES, ZERO-PAD
0x02B9   Symmetric Key Import2 - Allow wrapping override keywords
0x00FD   Symmetric Key Import2 - AES,PKOAEP2
0x0329   Symmetric Key Import2 - AESKW
0x02B4   Symmetric Key Import2 - AESKWCV
0x00F4   Symmetric Key Import2 - HMAC,PKOAEP2
0x0090   Symmetric Key Token Change - RTCMK
0x00F1   Symmetric Key Token Change2 - RTCMK
0x010B   SET Block Compose
0x010C   SET Block Decompose
0x0121   SET Block Decompose - PIN Extension IPINENC
0x0122   SET Block Decompose - PIN Extension OPINENC
0x0291   Transaction Validation - Generate
0x0292   Transaction Validation - Verify CSC-3
0x0293   Transaction Validation - Verify CSC-4
0x0294   Transaction Validation - Verify CSC-5
0x0310   Trusted Block Create - Activate an inactive block
0x030F   Trusted Block Create - Create Block in inactive form
0x0158   TR31 Export - Permit any CCA key if INCL-CV is specified
0x014D   TR31 Export - Permit version A TR-31 key blocks
0x014E   TR31 Export - Permit version B TR-31 key blocks
0x014F   TR31 Export - Permit version C TR-31 key blocks
0x0184   TR31 Export - Permit DATA to C0:G/C
0x0186   TR31 Export - Permit DATA to D0:B
0x01AB   TR31 Export - Permit DKYGENKY:DKYL0+DALL to E4
0x01AF   TR31 Export - Permit DKYGENKY:DKYL0+DALL to E5
0x01AA   TR31 Export - Permit DKYGENKY:DKYL0+DDATA to E4
```

Figure 107. CCA Domain Role Display panel - part 4



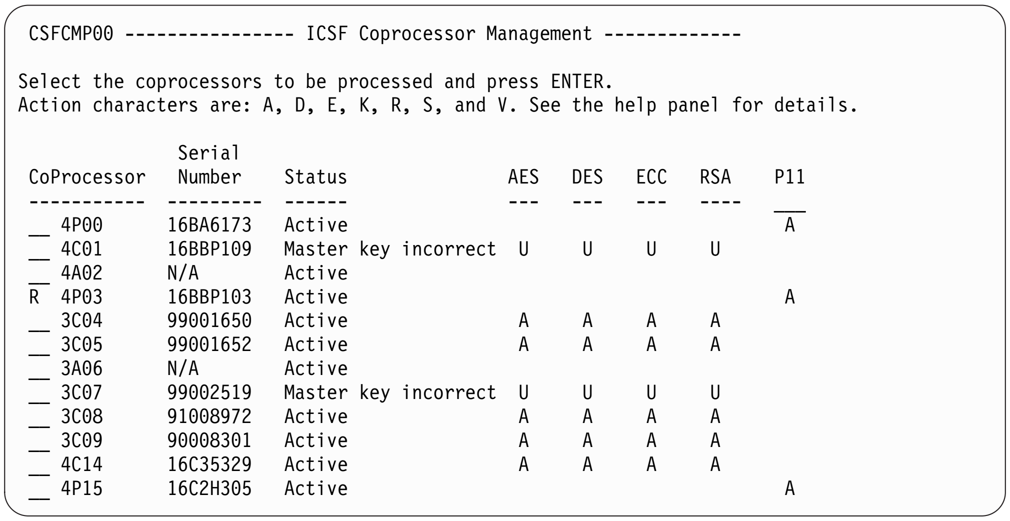
```
0x0185   TR31 Export - Permit ENCIPHER/DECIPHER/CIPHER to D0:E/D/B
0x0192   TR31 Export - Permit IPINENC to P0:D
0x0180   TR31 Export - Permit KEYGENKY:DUKPT to B0
0x018D   TR31 Export - Permit MAC/DATA/DATAM to M1:G/C
0x018F   TR31 Export - Permit MAC/DATA/DATAM to M3:G/C
0x0183   TR31 Export - Permit MAC/MACVER:ANY-MAC to C0:G/C/V
0x018C   TR31 Export - Permit MACVER/DATAMV to M0:V
0x018E   TR31 Export - Permit MACVER/DATAMV to M1:V
0x0190   TR31 Export - Permit MACVER/DATAMV to M3:V
0x0191   TR31 Export - Permit OPINENC to P0:E
0x0196   TR31 Export - Permit PINGEN:NO-SPEC/IBM-PIN/IBM-PINO to V1
0x0198   TR31 Export - Permit PINGEN:NO-SPEC/VISA-PVV to V2
0x0195   TR31 Export - Permit PINVER:NO-SPEC/IBM-PIN/IBM-PINO to V1
0x0197   TR31 Export - Permit PINVER:NO-SPEC/VISA-PVV to V2
0x0153   TR31 Import - Permit override of default wrapping method
0x0150   TR31 Import - Permit version A TR-31 key blocks
0x0151   TR31 Import - Permit version B TR-31 key blocks
0x0152   TR31 Import - Permit version C TR-31 key blocks
0x0178   TR31 Import - Permit E4 to DKYGENKY:DKYL0+DDATA
0x0164   TR31 Import - Permit M0/M1/M3 to MAC/MACVER:ANY-MAC
0x0166   TR31 Import - Permit P0:D to IPINENC
0x0165   TR31 Import - Permit P0:E to OPINENC
0x0169   TR31 Import - Permit V1 to PINGEN:IBM-PIN/IBM-PINO
0x016A   TR31 Import - Permit V1 to PINVER:IBM-PIN/IBM-PINO
0x016B   TR31 Import - Permit V2 to PINGEN:VISA-PVV
0x016C   TR31 Import - Permit V2 to PINVER:VISA-PVV
0x01C8   Unique Key Derive
0x01CA   Unique Key Derive - Override default wrapping
0x00E1   DUKPT - PIN Verify, PIN Translate
0x00DF   VISA CVV Generate
0x00E0   VISA CVV Verify
******************************* Bottom of data ********************************
```

Figure 108. CCA Domain Role Display panel - part 5

Displaying the EP11 domain roles

Use the ICSF panels to display the enabled access control points for the Enterprise PKCS #11 coprocessor. All the access control points enabled will be listed.

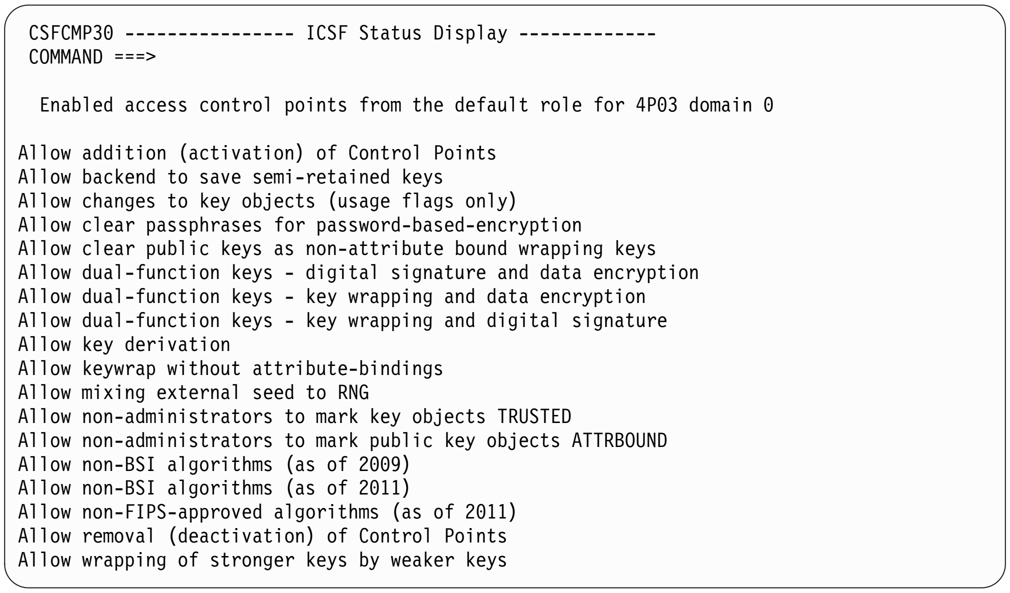
1. Select option 1, COPROCESSOR MGMT, on the "ICSF Primary Menu panel" on page 387.
2. The Coprocessor Management panel appears. Refer to Figure 109 on page 301.



```
CSFCMP00 ---------------- ICSF Coprocessor Management -------------

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

               Serial
CoProcessor   Number    Status              AES   DES   ECC   RSA    P11
-----------  ---------  ------              ---   ---   ---   ----   ___
__ 4P00      16BA6173   Active                                        A
__ 4C01      16BBP109   Master key incorrect  U     U     U     U
__ 4A02      N/A        Active
R  4P03      16BBP103   Active                                        A
__ 3C04      99001650   Active               A     A     A     A
__ 3C05      99001652   Active               A     A     A     A
__ 3A06      N/A        Active
__ 3C07      99002519   Master key incorrect  U     U     U     U
__ 3C08      91008972   Active               A     A     A     A
__ 3C09      90008301   Active               A     A     A     A
__ 4C14      16C35329   Active               A     A     A     A
__ 4P15      16C2H305   Active                                        A
```

Figure 109. Coprocessor Management Panel

3. Select the desired coprocessor by entering an 'R' to the left of the coprocessor. Press enter and the Status Display panel appears (Figure 110).



```
 CSFCMP30 ---------------- ICSF Status Display -------------
 COMMAND ===>

  Enabled access control points from the default role for 4P03 domain 0

Allow addition (activation) of Control Points
Allow backend to save semi-retained keys
Allow changes to key objects (usage flags only)
Allow clear passphrases for password-based-encryption
Allow clear public keys as non-attribute bound wrapping keys
Allow dual-function keys - digital signature and data encryption
Allow dual-function keys - key wrapping and data encryption
Allow dual-function keys - key wrapping and digital signature
Allow key derivation
Allow keywrap without attribute-bindings
Allow mixing external seed to RNG
Allow non-administrators to mark key objects TRUSTED
Allow non-administrators to mark public key objects ATTRBOUND
Allow non-BSI algorithms (as of 2009)
Allow non-BSI algorithms (as of 2011)
Allow non-FIPS-approved algorithms (as of 2011)
Allow removal (deactivation) of Control Points
Allow wrapping of stronger keys by weaker keys
```

Figure 110. CSFCMP30 — ICSF - Status Display

For the Access Control Points that are available on the Enterprise PKCS #11 coprocessor, see PKCS #11 Coprocessor Access Control Points in *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

Displaying installation exits

ICSF provides invocation points where you can use installation exits to perform processing that is specific to your installation. For example, ICSF provides a preprocessing and postprocessing exit invocation for each ICSF callable service. You can write and define an exit to set return codes at postprocessing of a callable service.

You must define each installation exit in the installation options data set. You define the ICSF name for the exit, the load module name of the exit, and the action ICSF takes if the exit fails. You can use the panels to view the ICSF name for each exit invocation. For a defined exit, you view the exit's load module name and fail options.

ICSF provides these types of exits:
- ICSF mainline exits
- Key generator utility program exit
- Callable services exits
- Cryptographic Key Data Set (CKDS) Conversion program exit
- Single-record, read-write exit
- CKDS retrieval exit
- Security exits

The mainline exits are called when you start and stop ICSF. The key generator utility program exit is called during key generator utility program processing. The callable services exits are called during each of the callable services. The CKDS conversion program exit is called during conversion of CUSP or PCF CKDS to ICSF CKDS format. The Single-record, read-write exit is called when an access to a single record is made to a disk copy of the CKDS. The security exits are called during initialization and stopping of ICSF, during a call to a callable service, and during access of a CKDS entry.

For a detailed description of the ICSF exits, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

To display installation exits:
1. Select option 3, OPSTAT, on the "ICSF Primary Menu panel" on page 387. The "CSFSOP00 — Installation Options panel" on page 392 appears.
2. Select option 2, Exits, on the "CSFSOP00 — Installation Options panel" on page 392.

   The first of the "CSFSOP30 — Installation Exits Display panel" on page 393s appears.

   The "CSFSOP30 — Installation Exits Display panel" on page 393 displays the ICSF name for all the possible installation exits your installation can write.
3. Scroll through the screens, to view all of the installation exits.

   The system programmer specified the exit identifier, the load-module-name, and the failure option for each exit your installation uses with the EXIT keyword in the installation options data set. On this panel, you can view information about any exit that is specified in the installation options data set. The exit identifier is the ICSF name for the exit.

Table 54 shows the names for some general ICSF exits. Table 55 and Table 56 on page 306 show the ICSF name for each callable service exit.

Table 54. General ICSF Exits and Exit Identifiers

General ICSF Exit	Exit Identifier
Conversion Exit	**CSFCONVX**
Cryptographic Key Data Set Retrieval Exit	**CSFCKDS**
Key Generator Utility Program Exit	**CSFKGUP**
Mainline Exits	**CSFEXIT2, CSFEXIT3, CSFEXIT4, CSFEXIT5**
Security Initialization Exit Point	**CSFESECI**
Security Key Exit Point	**CSFESECK**
Security Service Exit Point	**CSFESECS**
Security Termination Exit Point	**CSFESECT**
Single-record, read-write Exit Point	**CSFSRRW**

Table 55. Callable Service and its Exit Identifier

Service	Exit Identifier
Authentication Parameter Generate	**CSFAPG**
Clear PIN Encrypt	**CSFCPE**
Clear PIN Generate Alternate	**CSFCPA**
Clear Key Import	**CSFCKI**
Cipher/Decipher	**CSFEDC**
Cipher Text Translate2	**CSFCTT2**
Cipher Text Translate2 (with ALET)	**CSFCTT3**
Control Vector Translate	**CSFCVT**
Cryptographic Variable Encipher	**CSFCVE**
CVV Key Combine	**CSFCKC**
Data Key Import	**CSFDKM**
Decode	**CSFDCO**
Decipher	**CSFDEC**
Decipher (with ALET)	**CSFDEC1**
Data Key Export	**CSFDKX**
Digital Signature Generate	**CSFDSG**
Digital Signature Verify	**CSFDSV**
Diversified Key Generate	**CSFDKG**
Diversified Key Generate2	**CSFDKG2**
DK Deterministic PIN Generate	**CSFDDPG**
DK Migrate PIN	**CSFDMP**
DK PAN Modify in Transaction	**CSFDPMT**
DK PAN Translate	**CSFDPT**
DK PIN Change	**CSFDPC**
DK PIN Verify	**CSFDPV**

Table 55. Callable Service and its Exit Identifier (continued)

Service	**Exit Identifier**
DK PRW Card Number Update	**CSFDPNU**
DK PRW CMAC Generate	**CSFDPCG**
DK Random PIN Generate	**CSFDRPG**
DK Regenerate PRW	**CSFDRP**
ECC Diffie-Hellman	**CSFEDH**
Encode	**CSFECO**
Encipher under Master Key	**CSFEMK**
Encipher	**CSFENC**
Encipher (with ALET)	**CSFENC1**
Encrypted PIN Generate	**CSFEPG**
FPE Decipher	**CSFFPED**
FPE Encipher	**CSFFPEE**
FPE Translate	**CSFFPET**
HMAC Generate	**CSFHMG**
HMAC Verify	**CSFHMV**
ICSF Multi-Purpose Service	**CSFMPS**
Key Data Set List	**CSFKDSL**
Key Data Set Metadata Read	**CSFKDMR**
Key Data Set Metadata Write	**CSFKDMW**
Key Export	**CSFKEX**
Key Generate	**CSFKGN**
Key Generate2	**CSFKGN2**
Key Import	**CSFKIM**
Key Part Import	**CSFKPI**
Key Part Import2	**CSFKPI2**
Key Record Create	**CSFKRC**
Key Record Create2	**CSFKRC2**
Key Record Delete	**CSFKRD**
Key Record Read	**CSFKRR**
Key Record Read2	**CSFKRR2**
Key Record Write	**CSFKRW**
Key Record Write2	**CSFKRW2**
Key Test	**CSFKYT**
Key Test2	**CSFKYT2**
Key Test Extended	**CSFKYTX**
Key Translate	**CSFKTR**
MAC Generate	**CSFMGN**
MAC Generate (with ALET)	**CSFMGN1**
MAC Generate2	**CSFMGN2**
MAC Generate2 (with ALET)	**CSFMGN3**

Table 55. Callable Service and its Exit Identifier (continued)

Service	**Exit Identifier**
MAC Verify	**CSFMVR**
MAC Verify (with ALET)	**CSFMVR1**
MAC Verify2	**CSFMVR2**
MAC Verify2 (with ALET)	**CSFMVR3**
MDC Generate	**CSFMDG**
MDC Generate (with ALET)	**CSFMDG1**
Multiple Clear Key Import	**CSFCKM**
Multiple Secure Key Import	**CSFSCKM**
One-Way Hash Generate	**CSFOWH**
One-Way Hash Generate (with ALET)	**CSFOWH1**
PCI Interface	**CSFPCI**
Recover PIN From Offset	**CSFPFO**
PIN Change/Unblock	**CSFPCU**
PIN Generate	**CSFPGN**
PIN Generate	**CSFPGN**
PIN Translate	**CSFPTR**
PIN Verify	**CSFPVR**
PKA Decrypt	**CSFPKD**
PKA Encrypt	**CSFPKE**
PKA Key Generate	**CSFPKG**
PKA Key Import	**CSFPKI**
PKA Key Token Change	**CSFPKTC**
PKA Key Translate	**CSFPKT**
PKDS Key Record Read2	**CSFPRR2**
PKDS Record Create	**CSFPKRC**
PKDS Record Delete	**CSFPKRD**
PKDS Record Read	**CSFPKRR**
PKDS Record Write	**CSFPKRW**
Prohibit Export	**CSFPEX**
Prohibit Export Extended	**CSFPEXX**
Random Number Generate	**CSFRNG**
Random Number Generate Long	**CSFRNGL**
Remote Key Export	**CSFRKX**
Restrict Key Attribute	**CSFRKA**
Retained Key Delete	**CSFRKD**
Retained Key List	**CSFRKL**
Secure Key Import	**CSFSKI**
Secure Key Import2	**CSFSKI2**
Secure Messaging for Keys	**CSFSKY**
Secure Messaging for PINs	**CSFSPN**

Table 55. Callable Service and its Exit Identifier (continued)

Service	Exit Identifier
SET Block Compose	**CSFSBC**
SET Block Decompose	**CSFSBD**
Symmetric Algorithm Decipher	**CSFSAD**
Symmetric Algorithm Encipher	**CSFSAE**
Symmetric Key Generate	**CSFSYG**
Symmetric Key Import	**CSFSYI**
Symmetric Key Import2	**CSFSYI2**
Symmetric Key Export	**CSFSYX**
Symmetric MAC Generate	**CSFSMG**
Symmetric MAC Generate (with ALET)	**CSFSMG1**
Symmetric MAC Verify	**CSFSMV**
Symmetric MAC Verify (with ALET)	**CSFSMV1**
Symmetric Key Export with Data	**CSFSXD**
Transaction Validation	**CSFTRV**
Trusted Block Create	**CSFTBC**
TR-31 Export	**CSFT31X**
TR-31 Import	**CSFT31I**
Unique Key Derive	**CSFUDK**
VISA CVV Service Generate	**CSFCSG**
VISA VISA CVV Service Verify	**CSFCSV**

Table 56. Compatibility Service and its Exit Identifier

Service	Exit Identifier
Encipher under Master Key	**CSFEMK**
CUSP/PCF GENKEY Service	**CSFGKC**
CUSP/PCF RETKEY Service	**CSFRTC**
Cipher/Decipher	**CSFEDC**

The load module name is the name of the module that contains the exit. The LOAD MODULE column on the panel lists the load module name for each exit. The OPTIONS column on this panel lists the action to occur if the exit fails.

4. To change the module name or failure option of an exit or add a new exit when viewing this panel, access the installation options data set. In the data set, change how you specified an exit or specify a new exit and restart ICSF.

Displaying installation-defined callable services

ICSF provides callable services to perform cryptographic functions. You can write a callable service to perform a function unique to your installation. In the installation options data set, you must define each installation-defined callable service. You specify a number to identify the service to ICSF, and you specify the load module that contains the service. You can use the panels to view the number and module name for each installation-defined callable service.

To run an installation-defined service, you must:

- Write the service.
- Define the service.
- Write a service stub and link it with your application program.

For more information about writing, defining, and running an installation-defined service, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

To display information about installation-defined callable services:

1. Select option 3, OPSTAT, on the “ICSF Primary Menu panel” on page 387.

   The Installation Options panel appears. Refer to Figure 111.

```
CSFSOP00 ----------- ICSF - Installation Options --------
OPTION ===> 3

Enter the number of the desired option above.

  1  OPTIONS  -  Display Installation Options
  2  EXITS    -  Display Installation exits and exit options
  3  SERVICES -  Display Installation Defined Services
```

Figure 111. Installation Options Panel

2. Select option 3, Services, on the Installation Options Status panel.

   The Installation Defined Services panel appears. Refer to Figure 112.

```
CSFSOP40 -------- ICSF - Installation Defined Services --- ROW 1 TO 8 OF 8
COMMAND ===>

  SERVICE NUMBER        INSTALLATION NAME
  --------------        -----------------

         1               SERVICE1
         3               SERVICE3
         5               SERVICE5
         6               SERVICE6
         8               SERVICE8
        11               SERVICEB
        13               SERVICED
****************************BOTTOM OF DATA*************************************
```

Figure 112. Installation-Defined Services Display Panel

   The system programmer used the SERVICE keyword in the installation options data set to specify the service-number, the load-module-name, and fail-option for each service. The service number identies the service to ICSF. The load-module-name identifies the module that contains the installation-defined service. The Installation Name column on the panel lists the load-module-name for each installation service.

   The panel displays the service number and the corresponding installation name for each installation-defined service that is specified in the installation options data set.

Note: If your installation does not have any installation-defined callable services and you select option 3, the message NO GENERIC SERVICES displays and you remain on the Installation Options panel.

At ICSF start up, you define an installation options data set that contains the options your installation wants to use. The options specify certain modes and conditions on your ICSF system. You specify the keyword and value for each option in the installation options data set. You specify the data set name in the startup procedure. When you start ICSF, the options become active.

Chapter 13. Managing User Defined Extensions

User Defined Extensions (UDX) support allows you to request implementation of a customized cryptographic callable service. This support is available with a special contract with IBM for the CCA coprocessors. User Defined Extensions are ICSF functions developed for your installation with the help of IBM Global Services. Contact IBM Global Services for any problems with UDX.

You must define your routine to ICSF in the Installation Options Data Set. For more detailed information on the Installation Options Data Set and the UDX keyword, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

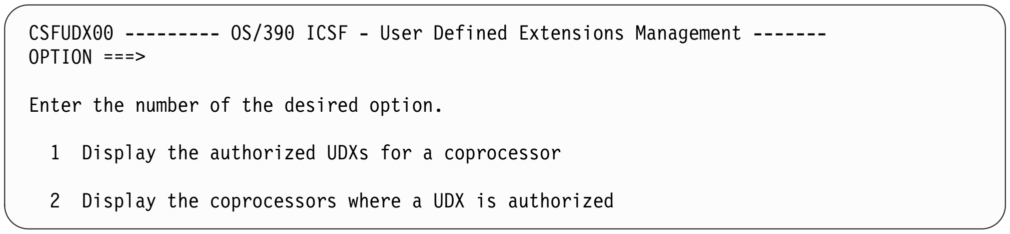
The UDX callable service load module is loaded during ICSF startup. Use the ICSF panels to perform UDX authorization processing.

You can perform these tasks:

- Display a list of UDX IDs of all authorized UDXs on a specific PCICC or CCA coprocessor.
- Display a list of all PCICCs and CCA coprocessors on which a specific UDX is authorized.

Select option 9, UDX MGMT, on the "ICSF Primary Menu panel" on page 387.

Once you have selected option 9, this panel is displayed:

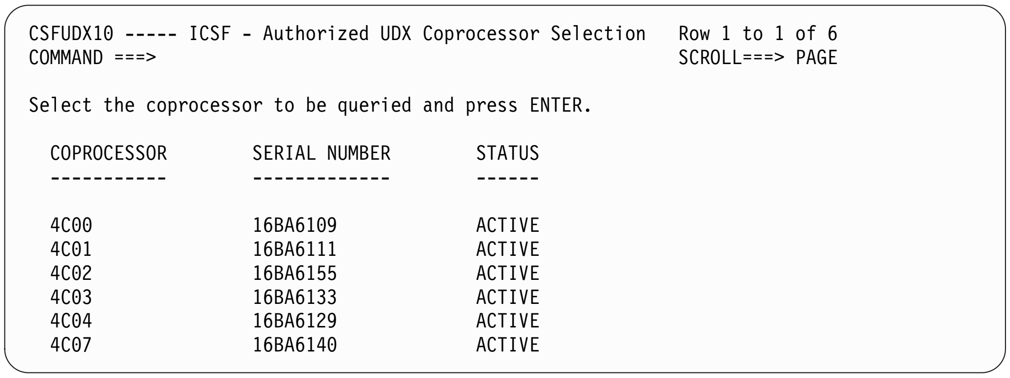


```
CSFUDX00 --------- OS/390 ICSF - User Defined Extensions Management -------
OPTION ===>

Enter the number of the desired option.

  1  Display the authorized UDXs for a coprocessor

  2  Display the coprocessors where a UDX is authorized
```

Figure 113. User Defined Extensions Management Panel

Display UDXs for a coprocessor

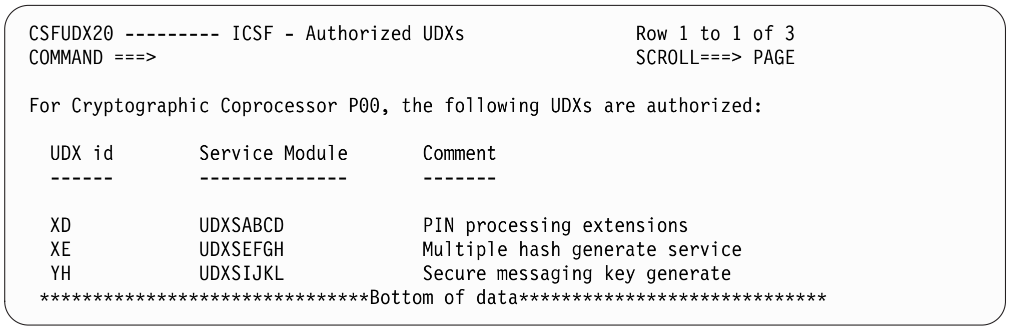
A panel similar to Figure 114 on page 310 is displayed when option 1 is selected. You will see a list of CCA coprocessors.



```
CSFUDX10 ----- ICSF - Authorized UDX Coprocessor Selection   Row 1 to 1 of 6
COMMAND ===>                                                 SCROLL===> PAGE

Select the coprocessor to be queried and press ENTER.

  COPROCESSOR        SERIAL NUMBER        STATUS
  -----------        -------------        ------

  4C00               16BA6109             ACTIVE
  4C01               16BA6111             ACTIVE
  4C02               16BA6155             ACTIVE
  4C03               16BA6133             ACTIVE
  4C04               16BA6129             ACTIVE
  4C07               16BA6140             ACTIVE
```

Figure 114. Authorized UDX Coprocessor Selection Panel

Select the coprocessor you wish to query. Use an **s** to select the coprocessor. Only one coprocessor can be selected. A panel similar to Figure 115 is displayed.



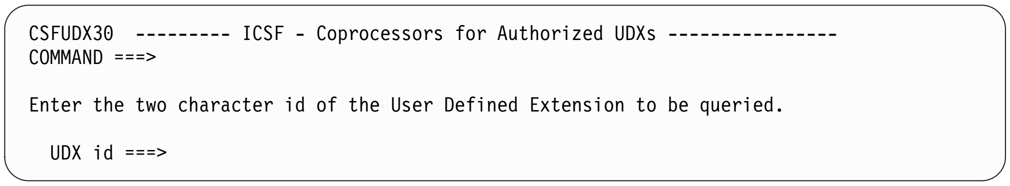
```
CSFUDX20 --------- ICSF - Authorized UDXs                Row 1 to 1 of 3
COMMAND ===>                                             SCROLL===> PAGE

For Cryptographic Coprocessor P00, the following UDXs are authorized:

  UDX id        Service Module       Comment
  ------        --------------       -------

  XD            UDXSABCD             PIN processing extensions
  XE            UDXSEFGH             Multiple hash generate service
  YH            UDXSIJKL             Secure messaging key generate
 *******************************Bottom of data******************************
```

Figure 115. Authorized UDXs Panel

This panel shows the authorized User Defined Extensions for the coprocessor selected. The UDX id is the two character code. The service module is the z/OS load module specified in the UDX keyword in the ICSF Installation Options Data Set. The comment is also specified in the UDX keyword.

Display coprocessors for a UDX

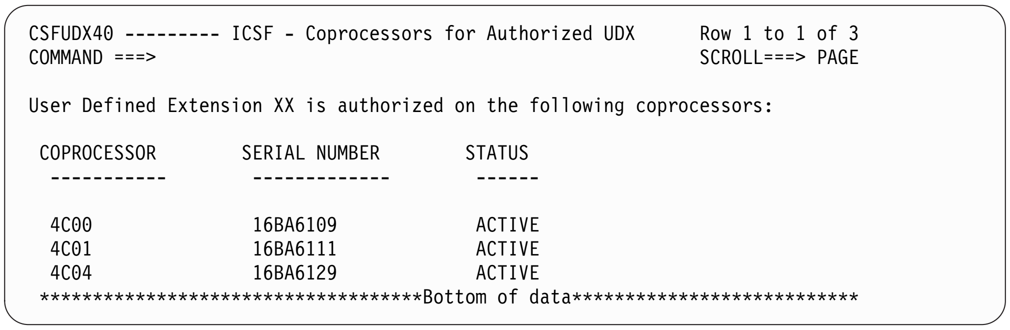
This panel is displayed when option 2 is selected from the User Defined Extensions Management Panel.



```
CSFUDX30  --------- ICSF - Coprocessors for Authorized UDXs ----------------
COMMAND ===>

Enter the two character id of the User Defined Extension to be queried.

  UDX id ===>
```

Figure 116. Coprocessors for Authorized UDXs Panel

Use this panel to specify the User Defined Extension id to be queried. A panel similar to Figure 117 appears.



```
CSFUDX40 --------- ICSF - Coprocessors for Authorized UDX      Row 1 to 1 of 3
COMMAND ===>                                                   SCROLL===> PAGE

User Defined Extension XX is authorized on the following coprocessors:

 COPROCESSOR        SERIAL NUMBER        STATUS
  -----------        -------------        ------

  4C00               16BA6109             ACTIVE
  4C01               16BA6111             ACTIVE
  4C04               16BA6129             ACTIVE
 *************************************Bottom of data***************************
```

Figure 117. Coprocessors for Authorized UDXs Panel

Chapter 14. Using the Utility Panels to Encode and Decode Data

Encoding data is enciphering data by using a clear key. Decoding data is deciphering data by using the same clear key that enciphered the data. You can use the utility panels to encode and decode data.

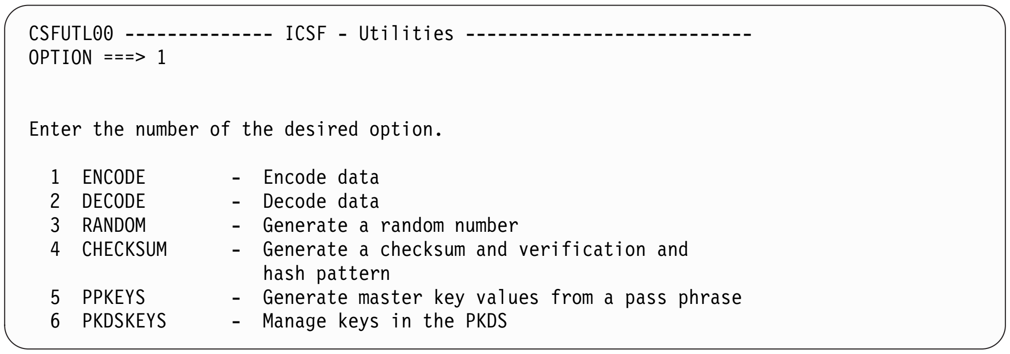
Note: ICSF must be active with a valid master key to use the encode and decode options. Encode and decode are available only on a DES-capable server or processor. CDMF-only systems cannot use encode and decode.

Steps for encoding data

To encode data:

1. Select option 5, UTILITY, on the "ICSF Primary Menu panel" on page 387.

   The Utilities panel appears. See Figure 118.

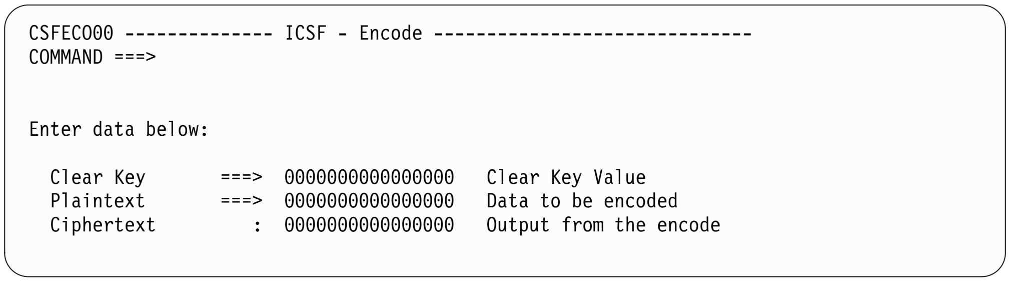


```
CSFUTL00 -------------- ICSF - Utilities ---------------------------
OPTION ===> 1


Enter the number of the desired option.

  1  ENCODE        -  Encode data
  2  DECODE        -  Decode data
  3  RANDOM        -  Generate a random number
  4  CHECKSUM      -  Generate a checksum and verification and
                      hash pattern
  5  PPKEYS        -  Generate master key values from a pass phrase
  6  PKDSKEYS      -  Manage keys in the PKDS
```

Figure 118. Selecting the Encode Option on the Utilities Panel

2. Select option 1, Encode, on this panel.

   The Encode panel appears. See Figure 119.



```
CSFECO00 -------------- ICSF - Encode ------------------------------
COMMAND ===>


Enter data below:

  Clear Key       ===>  0000000000000000   Clear Key Value
  Plaintext       ===>  0000000000000000   Data to be encoded
  Ciphertext         :  0000000000000000   Output from the encode
```

Figure 119. Encode Panel

3. In the Clear Key field, enter the clear value of the key you want ICSF to use to encode the data.

4. In the Plaintext field, enter the data in hexadecimal form that you want ICSF to encode.
5. Press ENTER.

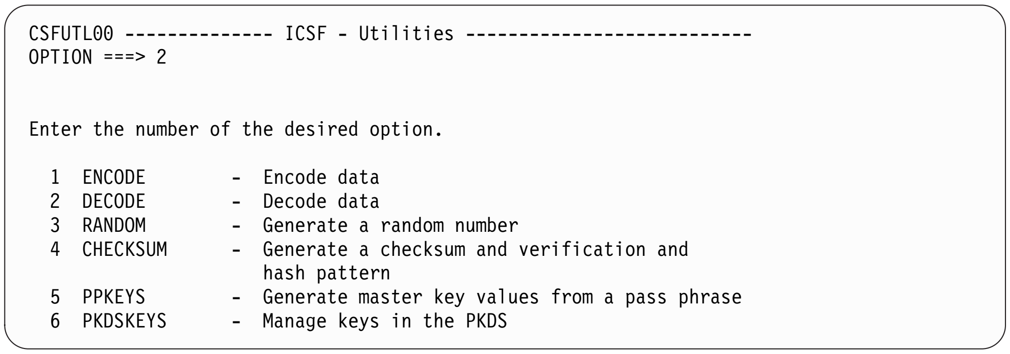
   ICSF uses the clear key and the DES algorithm to encode the data. The encoded data is displayed in the Ciphertext field.
6. Press END to return to the Utilities panel.
7. Press END to return to the Primary Option panel.

Steps for decoding data

To decode data:

1. Select option 5, UTILITY, on the Primary Option panel and press ENTER.

   The Utilities panel appears. See Figure 120.

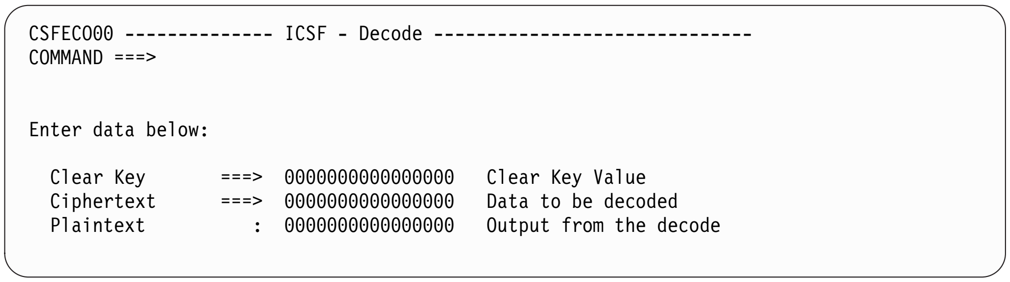


```
CSFUTL00 -------------- ICSF - Utilities ---------------------------
OPTION ===> 2


Enter the number of the desired option.

  1  ENCODE        -  Encode data
  2  DECODE        -  Decode data
  3  RANDOM        -  Generate a random number
  4  CHECKSUM      -  Generate a checksum and verification and
                      hash pattern
  5  PPKEYS        -  Generate master key values from a pass phrase
  6  PKDSKEYS      -  Manage keys in the PKDS
```

Figure 120. Selecting the Decode Option on the Utilities Panel

2. Select option 2, Decode, on this panel.

   The Decode panel appears. See Figure 121.



```
CSFECO00 -------------- ICSF - Decode ------------------------------
COMMAND ===>


Enter data below:

  Clear Key       ===>  0000000000000000   Clear Key Value
  Ciphertext      ===>  0000000000000000   Data to be decoded
  Plaintext          :  0000000000000000   Output from the decode
```

Figure 121. Decode Panel

3. In the Clear Key field, enter the clear value of the key you want ICSF to use to decode the data. This needs to be the same key value that was used to encode the data.
4. In the Ciphertext field, enter the data in hexadecimal form that you want ICSF to decode.
5. Press ENTER.

   ICSF uses the clear key and the DES algorithm to decode the data. The decoded data is displayed in the Plaintext field.

6. Press END to return to the Utilities panel.
7. Press END to return to the Primary Option panel.

Chapter 15. Using the utility panels to manage keys in the PKDS

The PKDSKEYS utility provides key management of records in the PKDS. The utility gives you the ability to:

- Generate an RSA or EC 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 that is stored in an MVS physically sequential data set.

These functions are intended for use with the Encryption Facility, but can be used for other purposes.

To use the full function of the ICSF PKDS Key Management panels, you must have a CCA coprocessor and the appropriate master keys must be active. The RSA master key must be active to generate RSA keys, and the ECC master key must be active to generate EC keys.

RACF protecting ICSF services used by the PKDS key management panels

ICSF uses these ICSF callable services to create or delete PKDS records and export or import RSA and EC keys to X.509 certificates:

CSNDKRR
Ensures that the specified PKDS label does not exist.

CSNDPKB
Builds the skeleton key token.

CSNDKRC
Creates the PKDS record.

CSNKRD
Deletes the PKDS record.

CSNDKRR
Reads the record from the PKDS.

CSNDPKX
Extracts only the public key from the record.

CSNBOWH
Hashes the to-be-signed portion of the generated certificate.

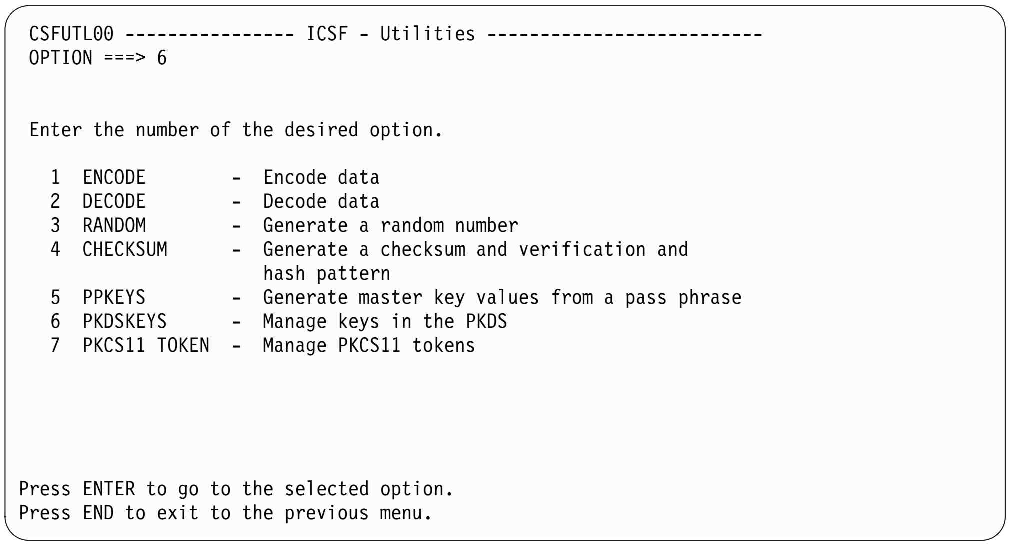
CSNDDSG
Signs the hash.

If you are using RACF or a similar security product, ensure that the security administrator authorizes ICSF to use these services and any cryptographic keys that are input. For information about ICSF callable services, see Introducing Symmetric Key Cryptography and Using Symmetric Key Callable Services in *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Managing keys in the PKDS

Follow these steps to manage keys in the PKDS:

1. Select option 6, PKDSKEYS, on the ICSF Utilities panel.



```
 CSFUTL00 ---------------- ICSF - Utilities --------------------------
 OPTION ===> 6


 Enter the number of the desired option.

   1  ENCODE        -  Encode data
   2  DECODE        -  Decode data
   3  RANDOM        -  Generate a random number
   4  CHECKSUM      -  Generate a checksum and verification and
                       hash pattern
   5  PPKEYS        -  Generate master key values from a pass phrase
   6  PKDSKEYS      -  Manage keys in the PKDS
   7  PKCS11 TOKEN  -  Manage PKCS11 tokens





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

Figure 122. Selecting the PKDSKEYS option on the ICSF Utilities Panel

2. When option 6, PKDSKEYS, is selected on the ICSF Utilities panel, the ICSF - PKDS Keys panel is presented.
   - If you are running on a z196, z114, or later zHardware, you have ECC support, and panel CSFPKY00 appears if the ECC master key is active.

```
CSFPKY00 ---------------- ICSF - PKDS Keys --------------------------

COMMAND ===>

    Enter the RSA record's label for the actions below
     ==>


    Select one of the following actions then press ENTER to process:

    -  Generate a new RSA or EC key pair record. Select one key type/size.
       RSA key bit length:  _ 512   _ 1024  _ 2048  _ 3072  _ 4096
       EC NIST Curve     :  _ p192  _ p224  _ p256  _ p384  _ p521
       EC Brainpool Curve:  _ p160  _ p192  _ p224  _ p256  _ p320  _ p384  _ p512
       Enter Private Key Name (optional)
        ==>

    -  Delete the existing public key or key pair PKDS record

    -  Export the PKDS record's public key to a certificate data set
       Enter the DSN  ===>
       Enter desired subject's common name (optional)
        CN=

    -  Create a PKDS public key record from an input certificate.
       Enter the DSN  ===>
```

Figure 123. ICSF PKDS Keys Panel

- Otherwise, panel CSFPKY10 appears.

```
CSFPKY10 ---------------- ICSF - PKDS Keys --------------------------

COMMAND ===>

  Enter the RSA record's label for the actions below
  ==>

  Select one of the following actions then press ENTER to process:

  - Generate a new RSA or EC key pair record. Select one key type/size.
    RSA key bit length:  _ 512   _ 1024  _ 2048  _ 3072  _ 4096
    Enter Private Key Name (optional)
     ==>

  - Delete the existing public key or key pair PKDS record

  - Export the PKDS record's public key to a certificate data set
    Enter the DSN  ===>
    Enter desired subject's common name (optional)
     CN=

  - Create a PKDS public key record from an input certificate.
    Enter the DSN ===>
```

Figure 124. ICSF PKDS Keys Panel

3. From this panel, you can manage RSA and EC key entries in the PKDS. To create a new record or manage an existing PKDS record, supply the PKDS key label and then select an action.

Supported actions:

- Generate a new RSA or EC public/private PKDS key pair record.

- Delete an existing key record.
- Export a public key to an X.509 certificate for importing elsewhere.
- Import a public key from an X.509 certificate received from elsewhere.

Generate a new RSA or EC public/private PKDS key pair record

The RSA key pair that is generated can be used to encrypt and recover archive data, recover encrypted data that is transmitted to you by another party, and sign and verify data. The EC key pair can be used to sign or verify data and to perform a key agreement operation with another party.

- The key type must be specified: RSA, EC NIST curve, or EC Brainpool curve.
- For RSA, the key length in bits must be specified (512, 1024, 2048, 3072, or 4096).
- For EC, the curve size must be specified.
- The private key name can also be specified, but is optional.
- Blank is the default for the private key name.
- Callable services:

  CSNDKRR
  Ensures that the specified PKDS label does not exist.

  CSNDPKB
  Builds the skeleton key token.

  CSNDPKG
  Generates the key pair.

  CSNDKRC
  Creates the PKDS record.

Note: The RSA public exponent that is used for all keys that are generated through this service is X'010001'.

Delete an existing key record

This service can be used to delete any PKDS record.

- If Delete is selected, the Delete PKDS Key Confirmation (CSFPKY0P) panel is displayed and forces you to confirm the delete.
- Callable services:

  CSNDKRD
  Deletes the PKDS record.

Note: If a public or private key pair record is deleted, any data encrypted with the private key is no longer recoverable.

Export a public key to an X.509 certificate for importation elsewhere

This service is used to encase the public half of a public/private key PKDS record into an X.509 digital certificate so that it can be sent to another party. You can receive data from another party that is enciphered under the public key, which you can recover by using the same PKDS record.

- The certificate that is created is stored in an MVS physical sequential data set.
- The output data set is created by the service with RECFM(V B).
- You must supply the data set name where the certificate is to be stored.
- The data set should not exist prior to export.

  – If the data set exists prior to export, its contents are destroyed and the data set reallocated new.
- The data set cannot be a PDS or PDS member.
- You can specify a value for the subject's common name in the certificate, if wanted.
  – If no value is specified, the PKDS record's label is used as the common name.
- Callable services:

  CSNDKRR
  Reads the record from the PKDS.

  CSNDPKX
  Extracts the public key from the record.

  CSNBOWH
  Hashes the to-be-signed portion of the generated certificate.

  CSNDDSG
  Signs the hash.

Notes:
1. The key record that is specified must be a public or private key pair record and must support signing.
2. The certificate has a 20-year validity period.
3. The certificate that is created is self-signed and DER encoded (binary).

Import a public key from an X.509 certificate received from elsewhere

This service is used to build a public PKDS key record from an X.509 digital certificate sent to you by another party. When completed, you can send the other party data that is enciphered under the public, which the other party can recover.
- The data set name that is supplied must contain the certificate
- The certificate must be a single DER encoded certificate.
- Base64 encoded certificates are not supported.
- The data set containing the certificate must be physical sequential with RECFM(V B).
- The data set cannot be a PDS or PDS member.
- Callable services:

  CSNDPKB
  Builds the public key token.

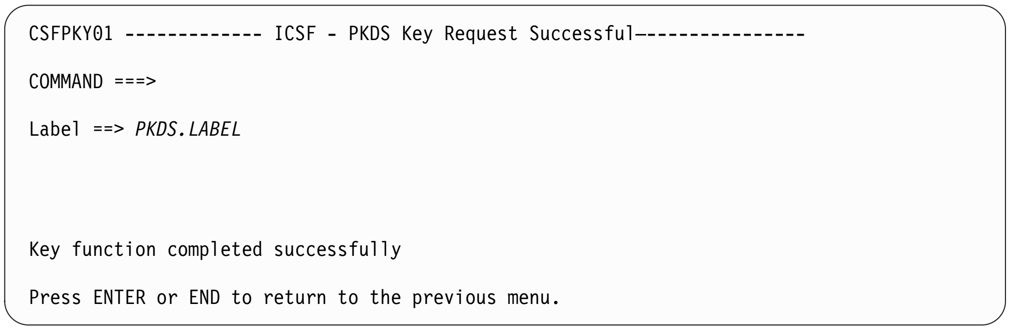
  CSNDKRC
  Creates the PKDS record.

Note: No signature check is performed on the certificate.

Processing Indicators

Success

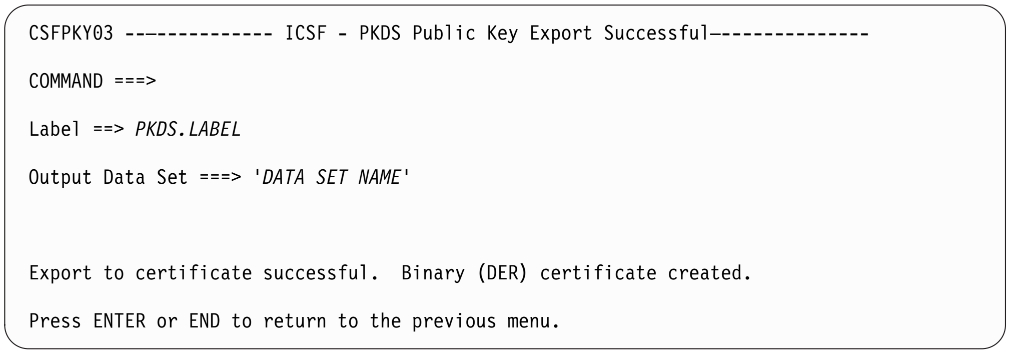
When Generate or Delete is specified and the function is successful, the PKDS Key Request Successful panel is presented:



```
CSFPKY01 ------------- ICSF - PKDS Key Request Successful----------------

COMMAND ===>

Label ==> PKDS.LABEL



Key function completed successfully

Press ENTER or END to return to the previous menu.
```

Figure 125. PKDS Key Request Successful

When Export is specified and the function is successful, the PKDS Public Key Export Successful panel is presented:



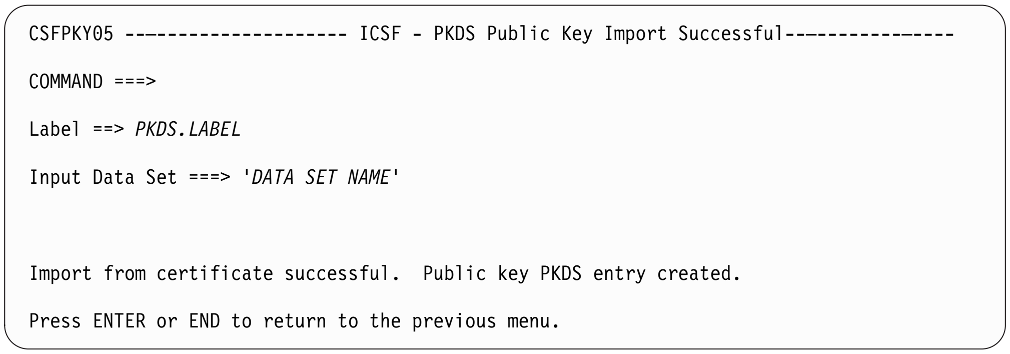
```
CSFPKY03 -------------- ICSF - PKDS Public Key Export Successful---------------

COMMAND ===>

Label ==> PKDS.LABEL

Output Data Set ===> 'DATA SET NAME'


Export to certificate successful.  Binary (DER) certificate created.

Press ENTER or END to return to the previous menu.
```

Figure 126. PKDS Public Key Export Successful

When these options are specified and the function is successful, these panels are generated:

- Generate or Delete - PKDS Key Request Successful panel (CSFPKY01)
- Export - PKDS Public Key Export Successful panel (CSFPKY03)
- Import - Public Key Import Successful panel (CSFPKY05)

When Import is specified and the function is successful, the PKDS Public Key Import Successful panel is presented:



```
CSFPKY05 --------------------- ICSF - PKDS Public Key Import Successful---------------

COMMAND ===>

Label ==> PKDS.LABEL

Input Data Set ===> 'DATA SET NAME'



Import from certificate successful.  Public key PKDS entry created.

Press ENTER or END to return to the previous menu.
```

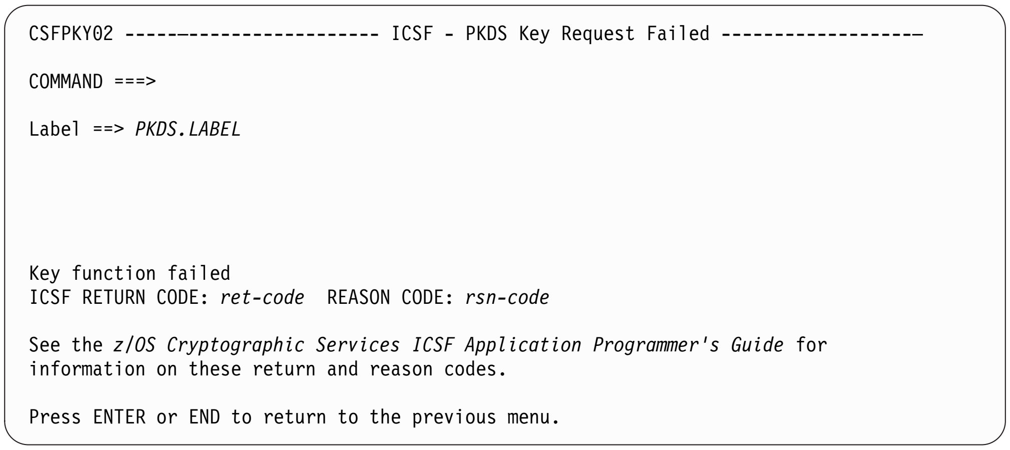
Figure 127. PKDS Public Key Import Successful

Failure

For the various functions, these expected errors generate an error message without presenting a new panel:

1. Panel input errors (for example, not specifying a PKDS label to work with).
2. ICSF not active.
3. Authorization failures (all functions).
4. Incorrect label syntax (all functions).
5. PKDS label exists (Generate and Import only).
6. PKDS label that is not found (Delete and Export only).
7. Specifying a PDS member (Import and Export only).
8. Cannot export a public key only PKDS record (Export only).

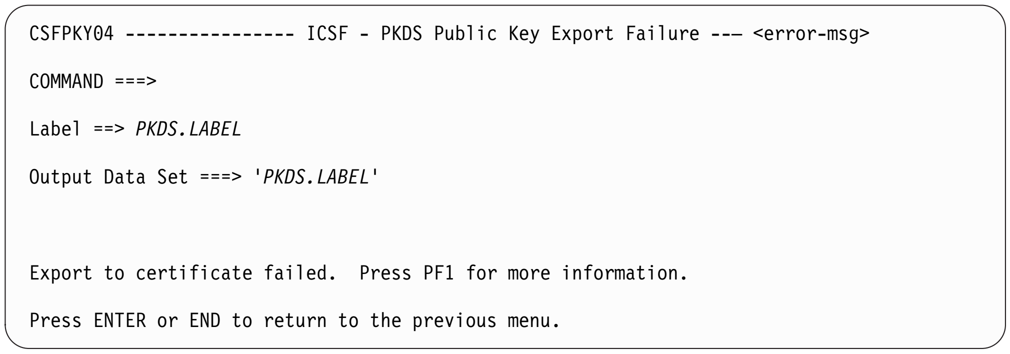
Unexpected ICSF callable service errors from any function cause the PKDS Key Request Failed panel to appear.



```
CSFPKY02 ------------------------ ICSF - PKDS Key Request Failed -------------------

COMMAND ===>

Label ==> PKDS.LABEL





Key function failed
ICSF RETURN CODE: ret-code  REASON CODE: rsn-code

See the z/OS Cryptographic Services ICSF Application Programmer's Guide for
information on these return and reason codes.

Press ENTER or END to return to the previous menu.
```

Figure 128. PKDS Key Request Failed

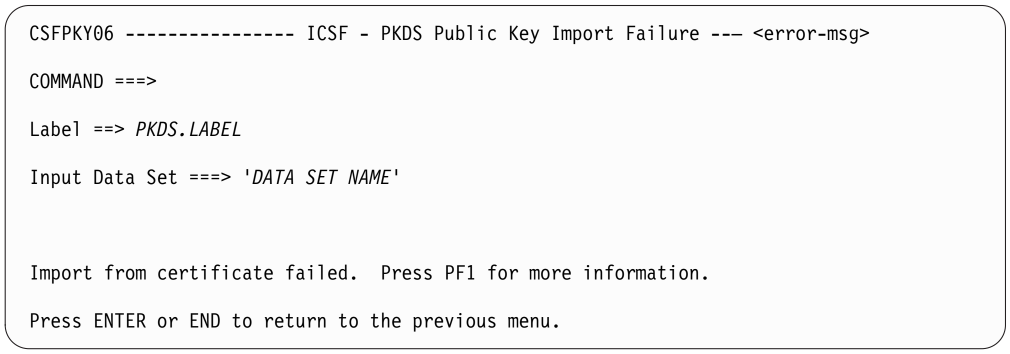
Non-ICSF related errors for export cause the PKDS Public Key Export Failure panel to appear.



```
CSFPKY04 ---------------- ICSF - PKDS Public Key Export Failure --– <error-msg>

COMMAND ===>

Label ==> PKDS.LABEL

Output Data Set ===> 'PKDS.LABEL'


Export to certificate failed.  Press PF1 for more information.

Press ENTER or END to return to the previous menu.
```

Figure 129. PKDS Public Key Export Failure

Non-ICSF related errors for import cause the PKDS Public Key Import Failure panel to appear.



```
CSFPKY06 ---------------- ICSF - PKDS Public Key Import Failure --– <error-msg>

COMMAND ===>

Label ==> PKDS.LABEL

Input Data Set ===> 'DATA SET NAME'


Import from certificate failed.  Press PF1 for more information.

Press ENTER or END to return to the previous menu.
```

Figure 130. PKDS Public Key Import Failure

Chapter 16. Using PKCS11 Token Browser Utility Panels

PKCS #11 is a standard set of programming interfaces for cryptographic functions. A subset of these functions is supported by ICSF. In the context of PKCS #11, a token is a representation of a cryptographic device, such as a smart card reader.

The PKCS11 token browser allows management of PKCS #11 tokens and objects in the TKDS. The PKCS11 token browser is option 7 PKCS11 TOKEN on the ICSF utilities panel. The user must have SAF authority to manage tokens and SAF authority to a token to manage the objects of a token (see “RACF Protecting ICSF Services used by the Token Browser Utility Panels”).

RACF Protecting ICSF Services used by the Token Browser Utility Panels

CRYPTOZ is a resource class defined in RACF in support of PKCS #11. Access to PKCS #11 tokens in ICSF is controlled by the CRYPTOZ class, with different access levels as well as a differentiation between standard users and security officers. For each token, there are two resources in the CRYPTOZ class for controlling access to tokens:

- The resource USER.*token-name* controls the access of the User role to the token
- The resource SO.*token-name* controls the access of the Security Officer (SO) role to the token.

A user's access level to each of these resources (read, update, or control) determines the user's access level to the token.

There are six possible token access levels. Three are defined by the PKCS #11 standard, and three are unique to z/OS®. The PKCS #11 token access levels are:

- User R/O: Allows the user to read the token including its private objects, but the user cannot create new token or session objects or alter existing ones.
- User R/W: Allows the user read/write access to the token object including its private objects.
- SO R/W: Allows the user to act as the security officer for the token and to read, create, and alter public objects on the token.

The token access levels unique to z/OS are:

- Weak SO: A security officer that can modify the CA certificates contained in a token but not initialize the token. (For example, a system administrator who determines the trust policy for all applications on the system.)
- Strong SO: A security officer that can add, generate or remove private objects in a token. (For example, a server administrator.)
- Weak User: A User that cannot change the trusted CAs contained in a token. (For example, to prevent an end-user from changing the trust policy of his or her token.)

Table 57 on page 326 shows how a user's access level to a token is derived from the user's access level to a resource in the SAF CRYPTOZ class.

Table 57. Token access levels. Token access levels

CRYPTOZ resource	SAF access level / READ	SAF access level / UPDATE	SAF access level / CONTROL
SO.*token-label*	Weak SO Can read, create, delete, modify, and use public objects	SO R/W Same ability as Weak SO plus can create and delete tokens	Strong SO Same ability as SO R/W plus can read but not use (see Note1) private objects; create, delete, and modify private objects
USER.*token-label*	User R/O Can read and use (see Note 1) public and private objects	Weak User Same ability as User R/O plus can create, delete, and modify private and public objects. Cannot add, delete, or modify certificate authority objects	User R/W Same ability as Weak User plus can add, delete, and modify certificate authority objects

Note:

1. "Use" is defined as any of these:
   - Performing any cryptographic operation involving the key object; for example C_Encrypt
   - Searching for key objects using sensitive search attributes
   - Retrieving sensitive key object attributes.

   The sensitive attribute for a secret key is CKA_VALUE. The sensitive attribute for the Diffie Hellman, DSA, and Elliptic Curve private key objects is CKA_VALUE. The sensitive attributes for RSA private key objects are CKA_PRIVATE_EXPONENT, CKA_PRIME_1, CKA_PRIME_2, CKA_EXPONENT_1, CKA_EXPONENT_2, and CKA_COEFFICIENT.
2. The CRYPTOZ resources can be defined as "RACF-DELEGATED" if required. For information about delegated resources, see *z/OS Security Server RACF Security Administrator's Guide*.
3. If the CSFSERV class is active, ICSF performs access control checks on the underlying callable services. The user must have READ access to the appropriate CSFSERV class resource. Table 58 lists the resources in the CSFSERV class for token services.
4. READ access is required for token management via RACDCERT or gskkyman command. To manage tokens through the token browser panels, you'll need READ access to services listed in Table 58.

Table 58. Resources in the CSFSERV class for token services

Name of resource	Service
CSF1GAV	Get object attributes
CSF1SAV	Update object attributes
CSF1TRC	Token or object creation
CSF1TRD	Token or object deletion
CSF1TRL	Token or object find

5. Although the use of generic profiles is permitted for the CRYPTOZ class, we recommend that you do not use a single generic profile to cover both the SO.*token-label* and USER.*token-label* resources. You should not do this, because another resource (FIPSEXEMPT.*token-label*, which is described in *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*) can be used to indicate whether compliance with the FIPS 140-2 standard is desired at the token level. Creating a profile that uses generic characters to match both the SO and USER portion of the resource names (for example *.*token-label*) will also inadvertently match the FIPSEXEMPT.token-label resource and can have unintended consequences.

Token browser panel utility

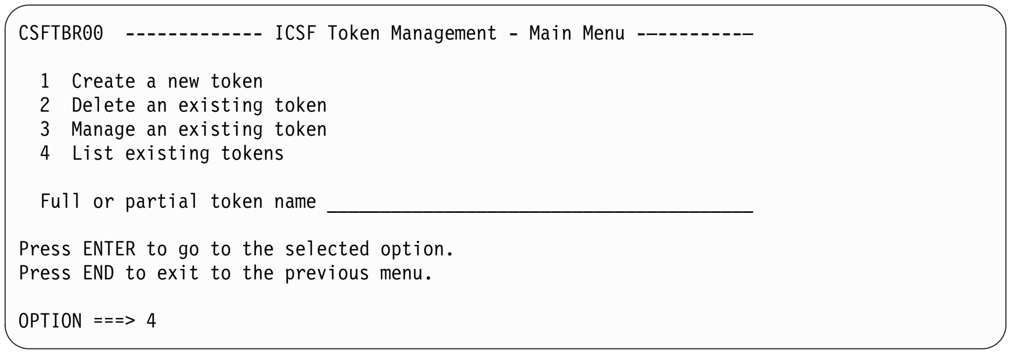
Steps to use the PKCS11 token browser panel utility

Follow these steps to use the PKCS11 token browser panel utility.

Select option 7, PKCS11 TOKEN, on the "CSFUTL00 — ICSF Utilities panel" on page 393.

Token Browser main panel

If option 7 is selected on the utilities panel, the ICSF Token Management - Main Menu is presented:

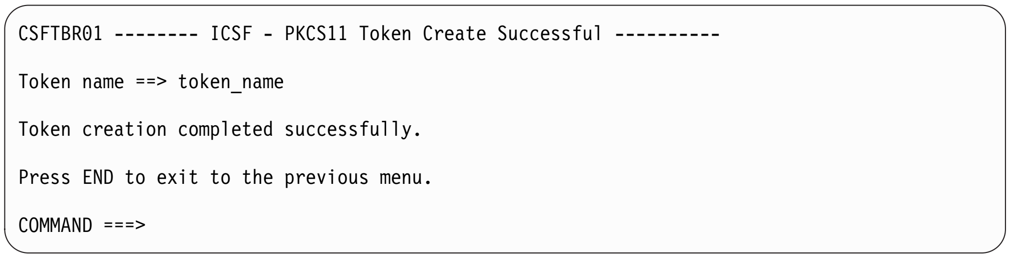


```
CSFTBR00  ------------- ICSF Token Management - Main Menu -----------

  1  Create a new token
  2  Delete an existing token
  3  Manage an existing token
  4  List existing tokens

  Full or partial token name ________________________________________

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

OPTION ===> 4
```

Figure 131. ICSF Token Management - Main Menu Panel

Token Create Successful

If option 1 is selected on the Main Menu panel, the ICSF - PKCS11 Token Create Successful menu is presented:

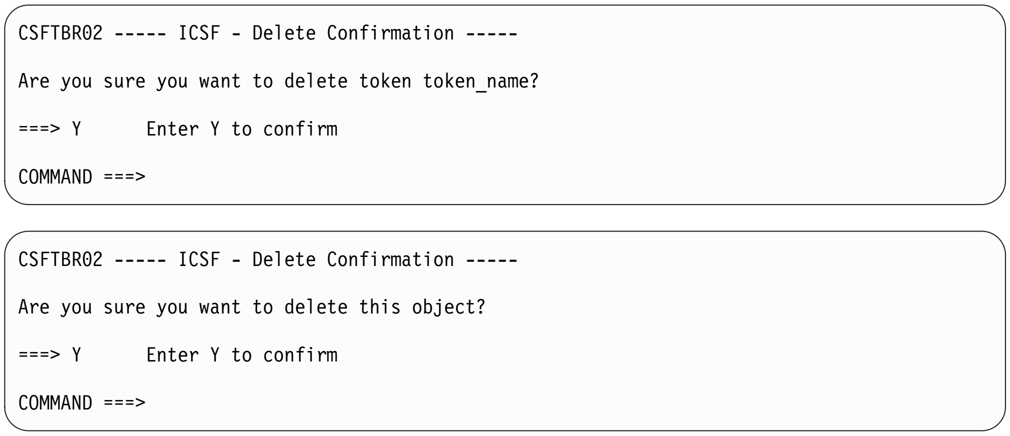


```
CSFTBR01 -------- ICSF - PKCS11 Token Create Successful ----------

Token name ==> token_name

Token creation completed successfully.

Press END to exit to the previous menu.

COMMAND ===>
```

Figure 132. ICSF Token Management - PKCS11 Token Create Successful panel

Token Delete Confirmation

If option 2 is selected on the Main Menu panel, the ICSF - Delete Confirmation menu is presented:



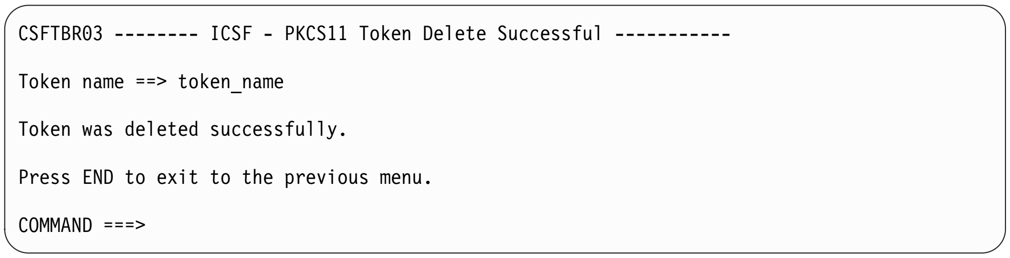
```
CSFTBR02 ----- ICSF - Delete Confirmation -----

Are you sure you want to delete token token_name?

===> Y      Enter Y to confirm

COMMAND ===>
```

```
CSFTBR02 ----- ICSF - Delete Confirmation -----

Are you sure you want to delete this object?

===> Y      Enter Y to confirm

COMMAND ===>
```

Figure 133. ICSF Token Management - PKCS11 Token Delete Confirmation panel

Token Delete Successful

If option Y is selected for delete token token_name on the ICSF - Delete Confirmation panel, the ICSF - PKCS11 Token Delete Successful menu is presented:



```
CSFTBR03 -------- ICSF - PKCS11 Token Delete Successful -----------

Token name ==> token_name

Token was deleted successfully.

Press END to exit to the previous menu.

COMMAND ===>
```

Figure 134. ICSF Token Management - PKCS11 Token Delete Successful panel

Object Delete Successful

If option Y is selected for delete this object on the ICSF - Delete Confirmation panel, the ICSF - PKCS11 Object Delete Successful menu is presented:

```
CSFTBR04 -------- ICSF - PKCS11 Object Delete Successful ---------

Object was deleted successfully.

Press END to exit to the previous menu.

COMMAND ===>
```

Figure 135. ICSF Token Management - PKCS11 Object Delete Successful panel

List Token panel

If option 4 is selected on the Main Menu panel, the ICSF Token Management - List Token menu is presented:

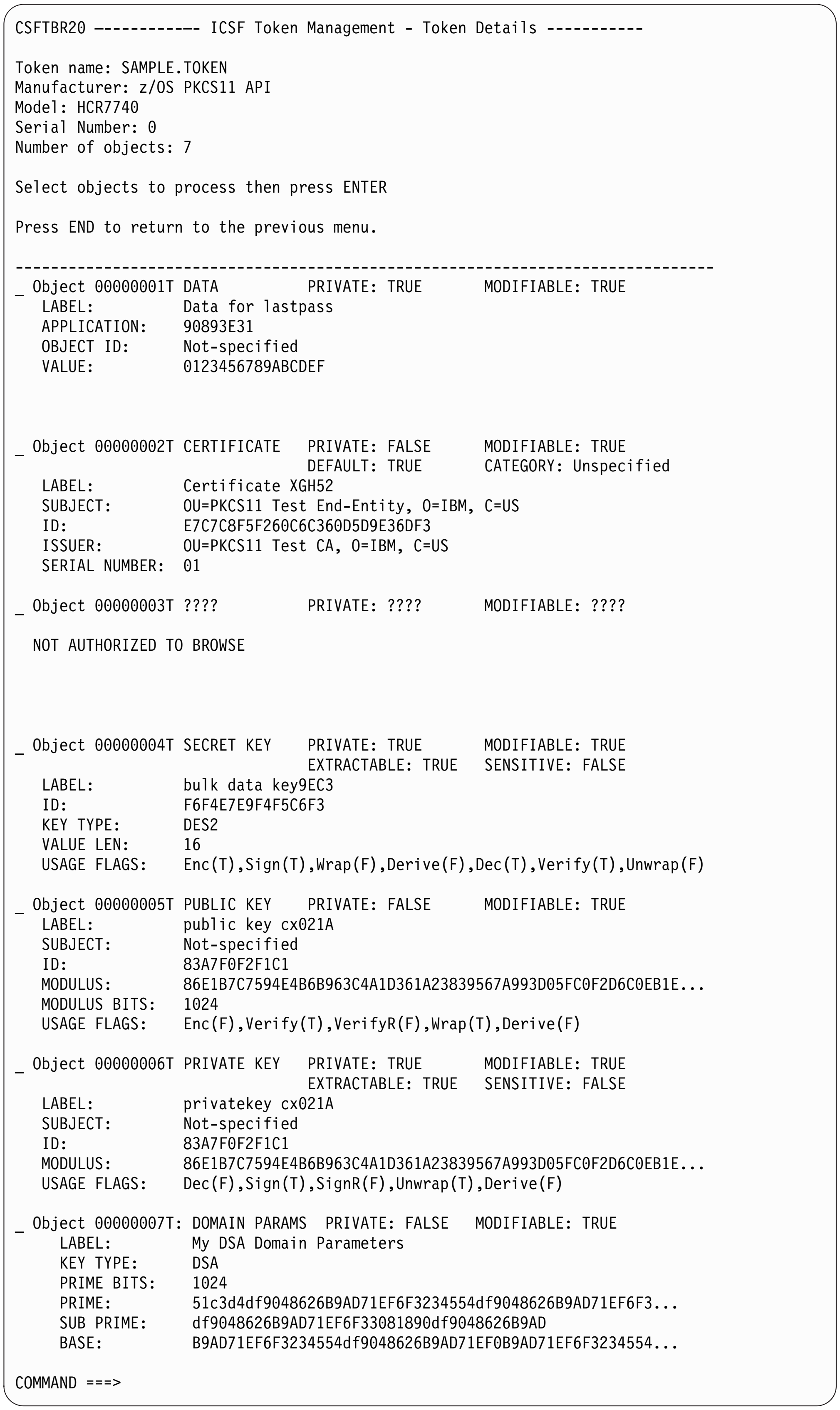
```
CSFTBR10 ------------ ICSF Token Management - List Tokens ---- Row 1 to 4 of 4

  Select a token to manage(M) or delete(D) then press ENTER

  Press END to return to the previous menu.

M SAMPLE.TOKEN
_ TOKEN.BOB
_ TOKEN.FRED
_ TOKEN.FRED.SECONDARY


COMMAND ===>
```

Figure 136. ICSF Token Management - List Token Panel

Note: Only tokens that you have authorization for are displayed.

Token Details panel

If manage (M) is selected on the List Tokens panel, the ICSF Token Management - Token Details menu is presented:

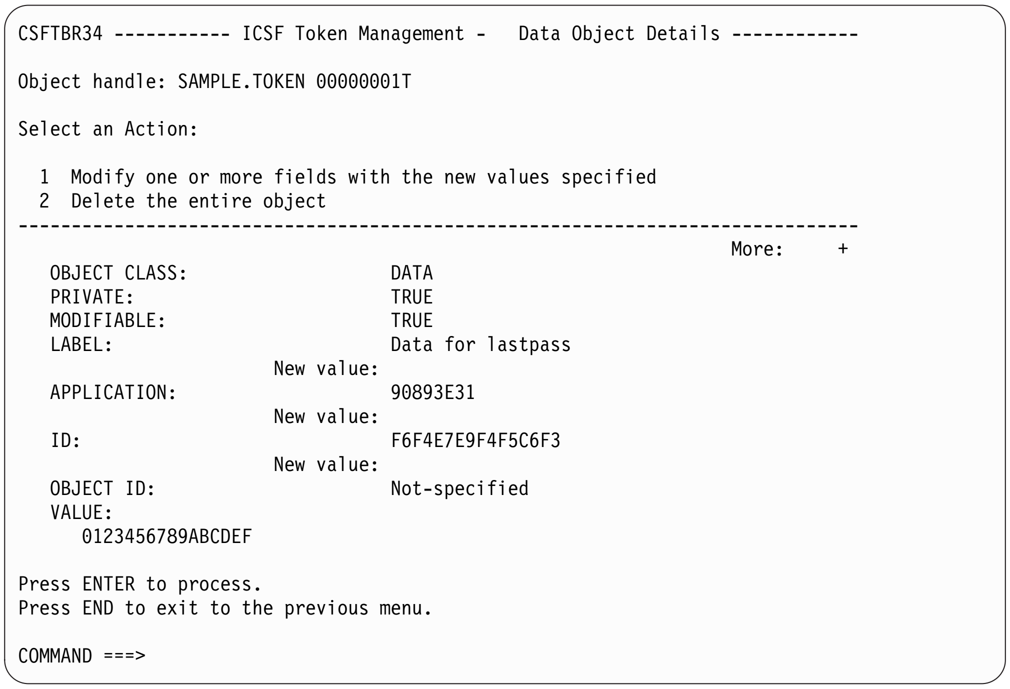


```
CSFTBR20 ------------ ICSF Token Management - Token Details -----------

Token name: SAMPLE.TOKEN
Manufacturer: z/OS PKCS11 API
Model: HCR7740
Serial Number: 0
Number of objects: 7

Select objects to process then press ENTER

Press END to return to the previous menu.

-------------------------------------------------------------------------------
_ Object 00000001T DATA          PRIVATE: TRUE       MODIFIABLE: TRUE
   LABEL:          Data for lastpass
   APPLICATION:    90893E31
   OBJECT ID:      Not-specified
   VALUE:          0123456789ABCDEF



_ Object 00000002T CERTIFICATE   PRIVATE: FALSE      MODIFIABLE: TRUE
                                 DEFAULT: TRUE       CATEGORY: Unspecified
   LABEL:          Certificate XGH52
   SUBJECT:        OU=PKCS11 Test End-Entity, O=IBM, C=US
   ID:             E7C7C8F5F260C6C360D5D9E36DF3
   ISSUER:         OU=PKCS11 Test CA, O=IBM, C=US
   SERIAL NUMBER:  01

_ Object 00000003T ????          PRIVATE: ????       MODIFIABLE: ????

  NOT AUTHORIZED TO BROWSE




_ Object 00000004T SECRET KEY    PRIVATE: TRUE       MODIFIABLE: TRUE
                                 EXTRACTABLE: TRUE   SENSITIVE: FALSE
   LABEL:          bulk data key9EC3
   ID:             F6F4E7E9F4F5C6F3
   KEY TYPE:       DES2
   VALUE LEN:      16
   USAGE FLAGS:    Enc(T),Sign(T),Wrap(F),Derive(F),Dec(T),Verify(T),Unwrap(F)

_ Object 00000005T PUBLIC KEY    PRIVATE: FALSE      MODIFIABLE: TRUE
   LABEL:          public key cx021A
   SUBJECT:        Not-specified
   ID:             83A7F0F2F1C1
   MODULUS:        86E1B7C7594E4B6B963C4A1D361A23839567A993D05FC0F2D6C0EB1E...
   MODULUS BITS:   1024
   USAGE FLAGS:    Enc(F),Verify(T),VerifyR(F),Wrap(T),Derive(F)

_ Object 00000006T PRIVATE KEY   PRIVATE: TRUE       MODIFIABLE: TRUE
                                 EXTRACTABLE: TRUE   SENSITIVE: FALSE
   LABEL:          privatekey cx021A
   SUBJECT:        Not-specified
   ID:             83A7F0F2F1C1
   MODULUS:        86E1B7C7594E4B6B963C4A1D361A23839567A993D05FC0F2D6C0EB1E...
   USAGE FLAGS:    Dec(F),Sign(T),SignR(F),Unwrap(T),Derive(F)

_ Object 00000007T: DOMAIN PARAMS  PRIVATE: FALSE   MODIFIABLE: TRUE
     LABEL:         My DSA Domain Parameters
     KEY TYPE:      DSA
     PRIME BITS:    1024
     PRIME:         51c3d4df9048626B9AD71EF6F3234554df9048626B9AD71EF6F3...
     SUB PRIME:     df9048626B9AD71EF6F33081890df9048626B9AD
     BASE:          B9AD71EF6F3234554df9048626B9AD71EF0B9AD71EF6F3234554...

COMMAND ===>
```

Figure 137. ICSF Token Management - Token Details panel

Data Object Details panel

If a data object is selected on the Token Details panel, the ICSF Token Management - Data Object Details menu is presented:

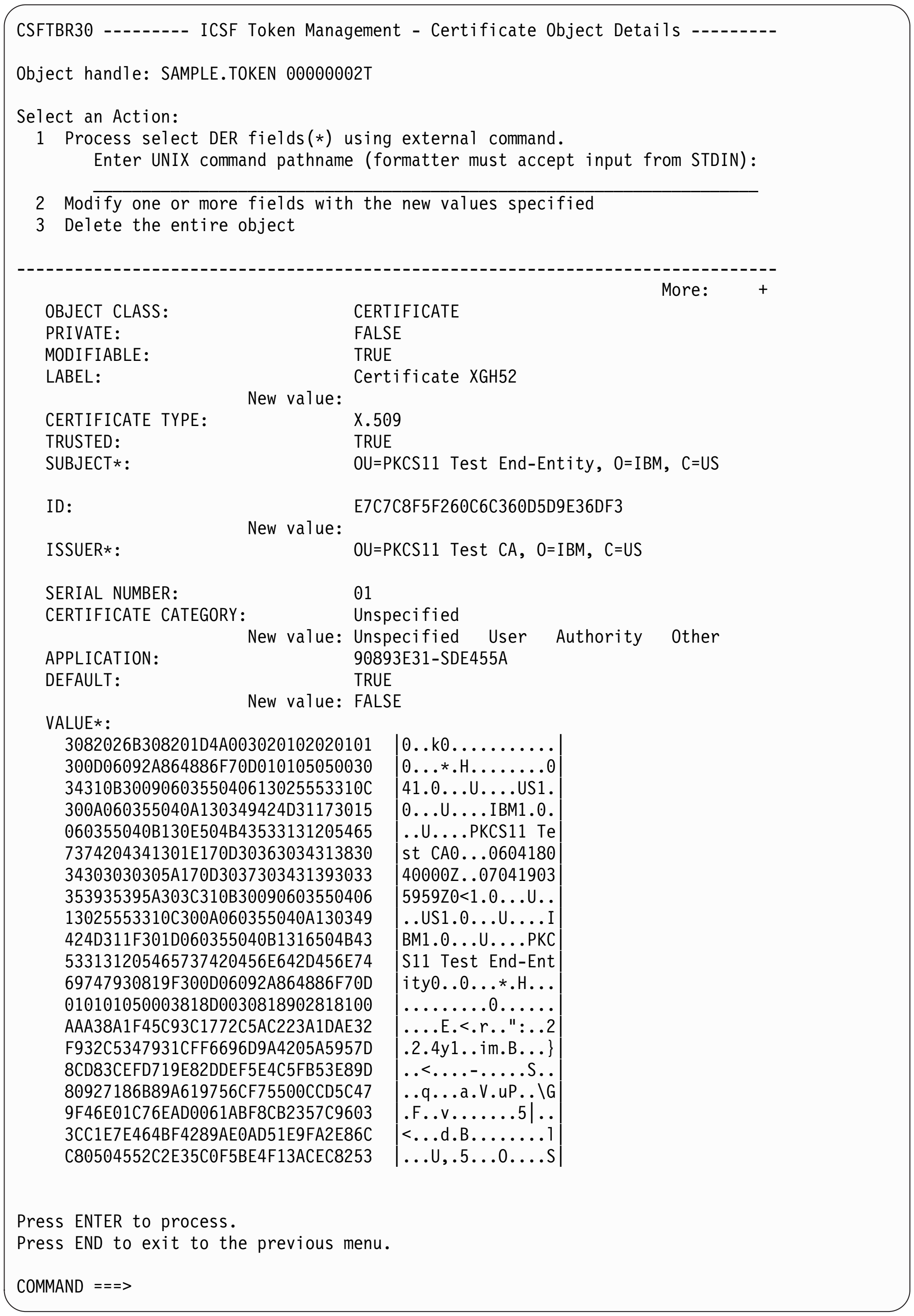


```
CSFTBR34 ----------- ICSF Token Management -   Data Object Details ------------

Object handle: SAMPLE.TOKEN 00000001T

Select an Action:

  1  Modify one or more fields with the new values specified
  2  Delete the entire object
-------------------------------------------------------------------------------
                                                                   More:     +
   OBJECT CLASS:                 DATA
   PRIVATE:                      TRUE
   MODIFIABLE:                   TRUE
   LABEL:                        Data for lastpass
                        New value:
   APPLICATION:                  90893E31
                        New value:
   ID:                           F6F4E7E9F4F5C6F3
                        New value:
   OBJECT ID:                    Not-specified
   VALUE:
      0123456789ABCDEF

Press ENTER to process.
Press END to exit to the previous menu.

COMMAND ===>
```

Figure 138. ICSF Token Management - Data Object Details panel

Certificate Object Details panel

If a certificate object is selected on the Token Details panel, the ICSF Token Management - Certificate Object Details menu is presented:

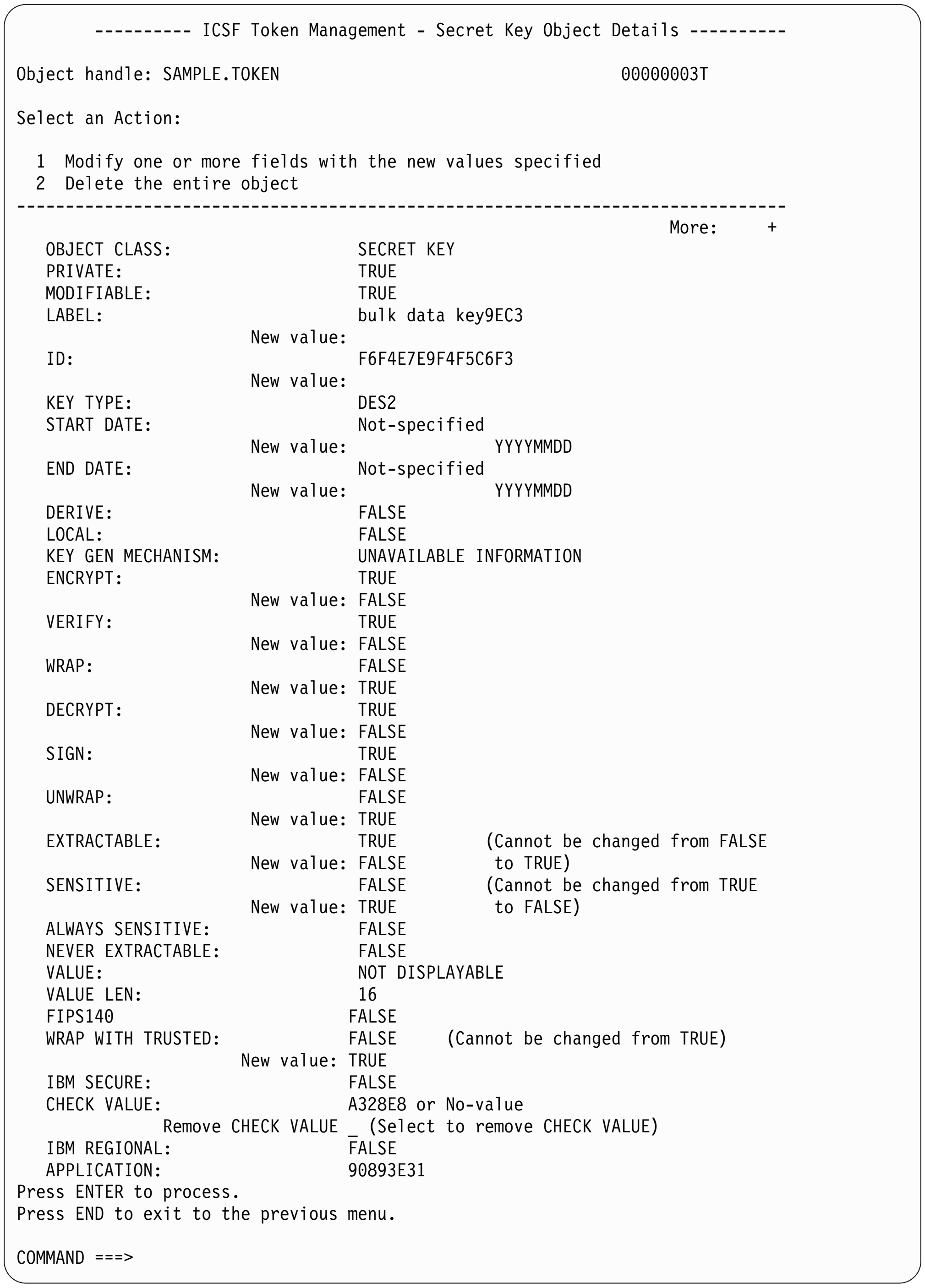


```
CSFTBR30 --------- ICSF Token Management - Certificate Object Details ---------

Object handle: SAMPLE.TOKEN 00000002T

Select an Action:
  1  Process select DER fields(*) using external command.
        Enter UNIX command pathname (formatter must accept input from STDIN):
        _____________________________________________________________________
  2  Modify one or more fields with the new values specified
  3  Delete the entire object

-------------------------------------------------------------------------------
                                                                  More:     +
   OBJECT CLASS:                   CERTIFICATE
   PRIVATE:                        FALSE
   MODIFIABLE:                     TRUE
   LABEL:                          Certificate XGH52
                       New value:
   CERTIFICATE TYPE:               X.509
   TRUSTED:                        TRUE
   SUBJECT*:                       OU=PKCS11 Test End-Entity, O=IBM, C=US

   ID:                             E7C7C8F5F260C6C360D5D9E36DF3
                       New value:
   ISSUER*:                        OU=PKCS11 Test CA, O=IBM, C=US

   SERIAL NUMBER:                  01
   CERTIFICATE CATEGORY:           Unspecified
                       New value: Unspecified   User   Authority   Other
   APPLICATION:                    90893E31-SDE455A
   DEFAULT:                        TRUE
                       New value: FALSE
   VALUE*:
     3082026B308201D4A003020102020101  |0..k0...........|
     300D06092A864886F70D010105050030  |0...*.H........0|
     34310B300906035504061302555331 0C |41.0...U....US1.|
     300A060355040A130349424D31173015  |0...U....IBM1.0.|
     060355040B130E504B43533131205465  |..U....PKCS11 Te|
     7374204341301E170D30363034313830  |st CA0...0604180|
     34303030305A170D3037303431393033  |40000Z..07041903|
     353935395A303C310B30090603550406  |5959Z0<1.0...U..|
     13025553310C300A060355040A130349  |..US1.0...U....I|
     424D311F301D060355040B1316504B43  |BM1.0...U....PKC|
     53313120546573742045 6E642D456E74 |S11 Test End-Ent|
     69747930819F300D06092A864886F70D  |ity0..0...*.H...|
     010101050003818D0030818902818100  |.........0......|
     AAA38A1F45C93C1772C5AC223A1DAE32  |....E.<.r..":..2|
     F932C5347931CFF6696D9A4205A5957D  |.2.4y1..im.B...}|
     8CD83CEFD719E82DDEF5E4C5FB53E89D  |..<....-.....S..|
     80927186B89A619756CF75500CCD5C47  |..q...a.V.uP..\G|
     9F46E01C76EAD0061ABF8CB2357C9603  |.F..v.......5|..|
     3CC1E7E464BF4289AE0AD51E9FA2E86C  |<...d.B........l|
     C80504552C2E35C0F5BE4F13ACEC8253  |...U,.5...O....S|


Press ENTER to process.
Press END to exit to the previous menu.

COMMAND ===>
```

Figure 139. ICSF Token Management - Certificate Object Details panel

Secret Key Object Details panel

If a secret key object is selected on the Token Details panel, the ICSF Token Management - Secret Key Object Details menu is presented:



```
      ---------- ICSF Token Management - Secret Key Object Details ----------

Object handle: SAMPLE.TOKEN                                   00000003T

Select an Action:

  1  Modify one or more fields with the new values specified
  2  Delete the entire object
-------------------------------------------------------------------------------
                                                                   More:     +
   OBJECT CLASS:                  SECRET KEY
   PRIVATE:                       TRUE
   MODIFIABLE:                    TRUE
   LABEL:                         bulk data key9EC3
                       New value:
   ID:                            F6F4E7E9F4F5C6F3
                       New value:
   KEY TYPE:                      DES2
   START DATE:                    Not-specified
                       New value:                YYYYMMDD
   END DATE:                      Not-specified
                       New value:                YYYYMMDD
   DERIVE:                        FALSE
   LOCAL:                         FALSE
   KEY GEN MECHANISM:             UNAVAILABLE INFORMATION
   ENCRYPT:                       TRUE
                       New value: FALSE
   VERIFY:                        TRUE
                       New value: FALSE
   WRAP:                          FALSE
                       New value: TRUE
   DECRYPT:                       TRUE
                       New value: FALSE
   SIGN:                          TRUE
                       New value: FALSE
   UNWRAP:                        FALSE
                       New value: TRUE
   EXTRACTABLE:                   TRUE         (Cannot be changed from FALSE
                       New value: FALSE         to TRUE)
   SENSITIVE:                     FALSE        (Cannot be changed from TRUE
                       New value: TRUE          to FALSE)
   ALWAYS SENSITIVE:              FALSE
   NEVER EXTRACTABLE:             FALSE
   VALUE:                         NOT DISPLAYABLE
   VALUE LEN:                     16
   FIPS140                       FALSE
   WRAP WITH TRUSTED:            FALSE     (Cannot be changed from TRUE)
                      New value: TRUE
   IBM SECURE:                   FALSE
   CHECK VALUE:                  A328E8 or No-value
            Remove CHECK VALUE _ (Select to remove CHECK VALUE)
   IBM REGIONAL:                 FALSE
   APPLICATION:                  90893E31
Press ENTER to process.
Press END to exit to the previous menu.

COMMAND ===>
```

Figure 140. ICSF Token Management - Secret Key Object Details panel

If IBM SECURE is TRUE, the panel will display these additional secure key attributes:

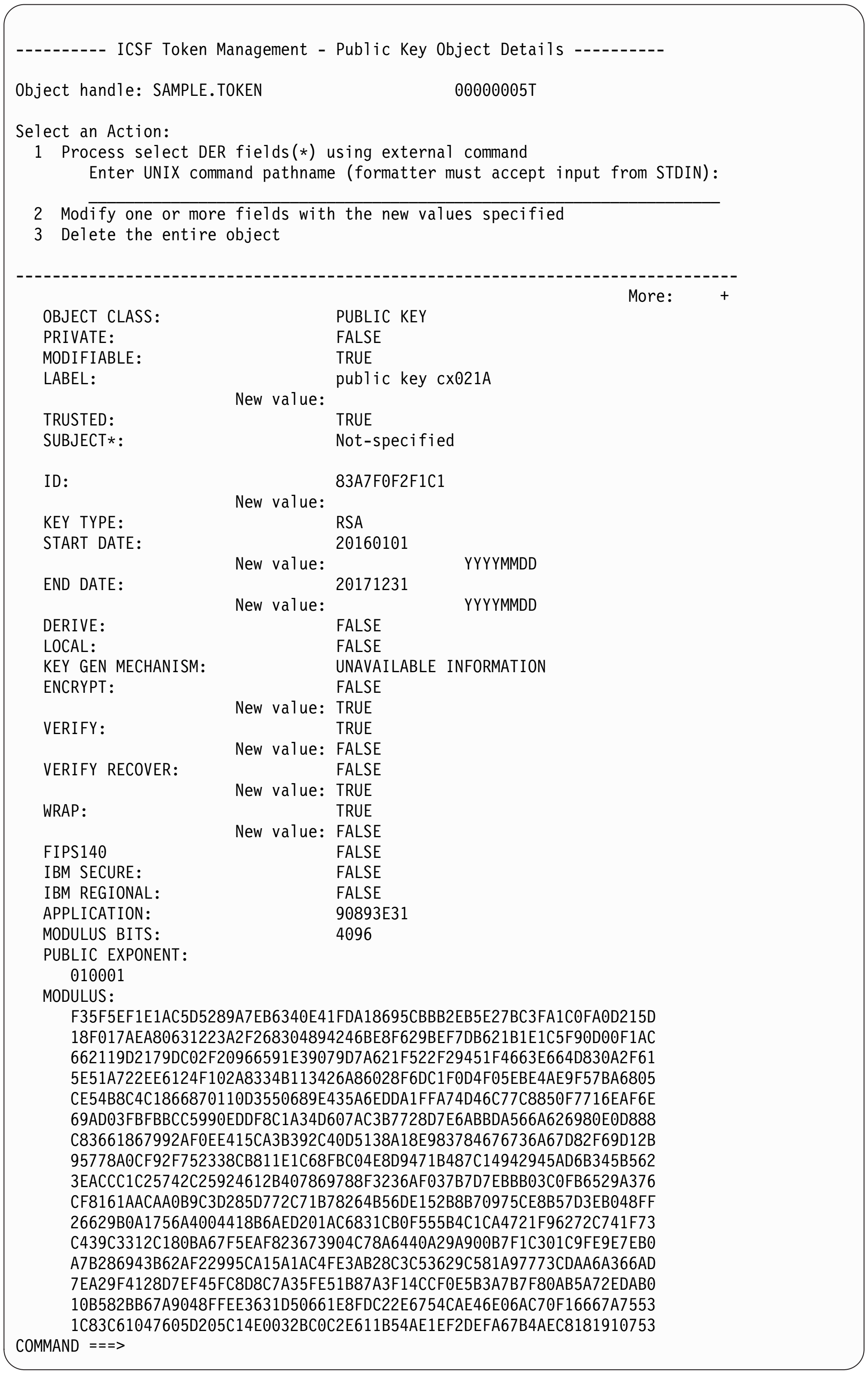
```
IBM ALWAYS SECURE:          TRUE
IBM ATTRBOUND:              FALSE
IBM CARD COMPLIANCE:        FIPS2009
       New Value: Set to current processor value ___
                  FIPS 2009 _  BSI2009 _  (Or select one or
                  FIPS 2011 _  BSI 2011_   more from list)
```

With regional cryptographic keys, if IBM SECURE is TRUE, the panel will display these additional secure key attributes:

```
IBM ALWAYS SECURE:          TRUE or FALSE
IBM ATTRBOUND:              FALSE
IBM CARD COMPLIANCE:        NOT APPLICABLE
```

Public Key Object Details panel

If a public key object is selected on the Token Details panel, the ICSF Token Management - Public Key Object Details panel is presented:



```
---------- ICSF Token Management - Public Key Object Details ----------

Object handle: SAMPLE.TOKEN                     00000005T

Select an Action:
  1  Process select DER fields(*) using external command
        Enter UNIX command pathname (formatter must accept input from STDIN):
        _____________________________________________________________________
  2  Modify one or more fields with the new values specified
  3  Delete the entire object

-------------------------------------------------------------------------------
                                                                   More:     +
   OBJECT CLASS:                   PUBLIC KEY
   PRIVATE:                        FALSE
   MODIFIABLE:                     TRUE
   LABEL:                          public key cx021A
                        New value:
   TRUSTED:                        TRUE
   SUBJECT*:                       Not-specified

   ID:                             83A7F0F2F1C1
                        New value:
   KEY TYPE:                       RSA
   START DATE:                     20160101
                        New value:               YYYYMMDD
   END DATE:                       20171231
                        New value:               YYYYMMDD
   DERIVE:                         FALSE
   LOCAL:                          FALSE
   KEY GEN MECHANISM:              UNAVAILABLE INFORMATION
   ENCRYPT:                        FALSE
                        New value: TRUE
   VERIFY:                         TRUE
                        New value: FALSE
   VERIFY RECOVER:                 FALSE
                        New value: TRUE
   WRAP:                           TRUE
                        New value: FALSE
   FIPS140                         FALSE
   IBM SECURE:                     FALSE
   IBM REGIONAL:                   FALSE
   APPLICATION:                    90893E31
   MODULUS BITS:                   4096
   PUBLIC EXPONENT:
      010001
   MODULUS:
      F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D
      18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC
      662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61
      5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805
      CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E
      69AD03FBFBBCC5990EDDF8C1A34D607AC3B7728D7E6ABBDA566A626980E0D888
      C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B
      95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562
      3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376
      CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF
      26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73
      C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0
      A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD
      7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0
      10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553
      1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753
COMMAND ===>
```

Figure 141. ICSF Token Management - Public Key Object Details panel

If IBM SECURE is TRUE then the panel will display these additional secure key attributes:

```
IBM ATTRBOUND:            FALSE
IBM CARD COMPLIANCE:      FIPS2009
       New Value: Set to current processor value ___
                  FIPS 2009 _   BSI2009 _  (Or select one or
                  FIPS 2011 _   BSI 2011_   more from list)
```

With regional cryptographic keys, IBM SECURE is TRUE so the panel will display these additional secure key attributes:

```
IBM ALWAYS SECURE:    TRUE or FALSE
IBM ATTRBOUND:        FALSE
IBM CARD COMPLIANCE:  NOT APPLICABLE
```

The format of the ICSF Token Management - Public Key Object Details panel will differ slightly depending on the type of key (RSA, DSA, Diffie-Hellman, Elliptic Curve, or SM2) selected.

Table 59. Information displayed in Public Key Object Details panel for RSA, DSA, Diffie-Hellman, Elliptic Curve, and SM2 keys

For this type of key:	**Identified in the panel's KEY TYPE field as:**	**The panel will contain fields for:**

RSA	RSA	The RSA modulus size, the public key exponent, and the RSA modulus. For example:
		`MODULUS BITS:                  4096`
		`PUBLIC EXPONENT:`
		`   010001`
		`MODULUS:`
		`   F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D`
		`   18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC`
		`   662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61`
		`   5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805`
		`   CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E`
		`   69AD03FBFBBCC5990EDDF8C1A34D607AC3B7728D7E6ABBDA566A626980E0D888`
		`   C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B`
		`   95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562`
		`   3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376`
		`   CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF`
		`   26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73`
		`   C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0`
		`   A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD`
		`   7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0`
		`   10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553`
		`   1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753`

DSA	DSA	The DSA prime *p*, subprime *q*, base *g*, and public value. For example:
		`PRIME:`
		`   2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A`
		`   ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33`
		`   C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9`
		`   4490C4837B5656776E9FFDA073EAED869B19F7E197970DBE5665E8F87F964C57`
		`SUBPRIME:`
		`   DF9048626B9AD71EF6F33081890DF9048626B9AD`
		`BASE:`
		`   F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A`
		`   D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE`
		`   1362CDDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1`
		`   1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40`
		`VALUE:`
		`   3992F874061239B0A0B2E52BDBC33237A1CEAB624B613AD91D23CDCCFC58D575`
		`   1CB5FE0364D37BF74721AA5473DD1ECC2E65B82138BE7103477C438B3486C548`
		`   0CCAD36D7882C9659CA32744E776DE894193953F6DF32C8AC3ACFC0A364A641A`
		`   A19B74BDC43E6EC84D8CB409B46A82D666A7F963D31A2CC897B971D378959509`

Table 59. Information displayed in Public Key Object Details panel for RSA, DSA, Diffie-Hellman, Elliptic Curve, and SM2 keys (continued)

For this type of key:	Identified in the panel's KEY TYPE field as:	The panel will contain fields for:

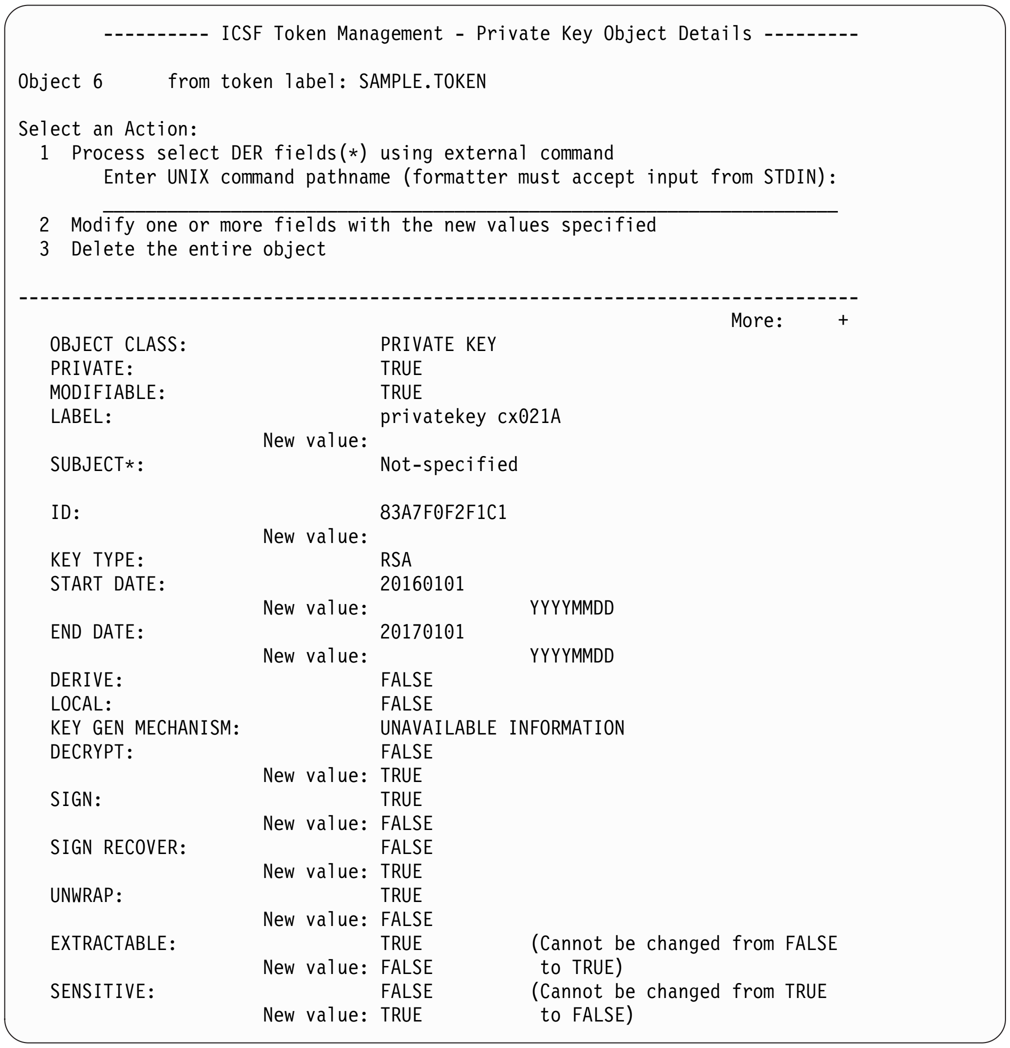
Diffie-Hellman	DH	The Diffie-Hellman prime *p*, base *g*, and public value. For example:
		`PRIME:`
		`F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D`
		`18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC`
		`662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61`
		`5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805`
		`CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E`
		`69AD03FBFBBCC5990EDDF8C1A34D607AC3B7728D7E6ABBDA566A626980E0D888`
		`C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B`
		`95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562`
		`BASE:`
		`3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376`
		`CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF`
		`26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73`
		`C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0`
		`A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD`
		`7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0`
		`10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553`
		`1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753`
		`VALUE:`
		`2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A`
		`ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33`
		`C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9`
		`4490C4837B5656776E9FFDA073EAED869B19F7E197970DBE5665E8F87F964C57`
		`F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A`
		`D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE`
		`1362CDDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1`
		`1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40`

Elliptic Curve	EC	The elliptic curve parameters and the elliptic curve point. For example:
		`_ EC PARAMS*:              Named Curve – secp521r1`
		`_ EC POINT*:`
		`3992F874061239B0A0B2E52BDBC33237A1CEAB624B613AD91D23CDCCFC58D575`
		`1CB5FE0364D37BF74721AA5473DD1ECC2E65B82138BE7103477C438B3486C548`
		`0CCAD36D7882C9659CA32744E776DE894193953F6DF32C8AC3ACFC0A364A641A`
		`A19B74BDC43E6EC84D8CB409B46A82D666A7F963D31A2CC897B971D378959509`
		`308189028181008B53`

SM2	SM2	The elliptic curve parameters and the elliptic curve point. For example:
		`_ EC PARAMS*:              Named Curve – GB-256`
		`_ EC POINT*:`
		`A29B10215E90DBC55B0FF96B38E2D31331085A5053F64CFF2E71F1F4D3F32505`
		`DB5DE3D9EFF524F640895130BB1D5E833201BE05323784C5772733D1518173FF`

Private Key Object Details panel

If a private key object is selected on the Token Details panel, the ICSF Token Management - Private Key Object Details panel is presented:



```
       ---------- ICSF Token Management - Private Key Object Details ---------

Object 6      from token label: SAMPLE.TOKEN

Select an Action:
  1  Process select DER fields(*) using external command
        Enter UNIX command pathname (formatter must accept input from STDIN):
        _____________________________________________________________________
  2  Modify one or more fields with the new values specified
  3  Delete the entire object

-------------------------------------------------------------------------------
                                                                More:     +
   OBJECT CLASS:                 PRIVATE KEY
   PRIVATE:                      TRUE
   MODIFIABLE:                   TRUE
   LABEL:                        privatekey cx021A
                      New value:
   SUBJECT*:                     Not-specified

   ID:                           83A7F0F2F1C1
                      New value:
   KEY TYPE:                     RSA
   START DATE:                   20160101
                      New value:              YYYYMMDD
   END DATE:                     20170101
                      New value:              YYYYMMDD
   DERIVE:                       FALSE
   LOCAL:                        FALSE
   KEY GEN MECHANISM:            UNAVAILABLE INFORMATION
   DECRYPT:                      FALSE
                      New value: TRUE
   SIGN:                         TRUE
                      New value: FALSE
   SIGN RECOVER:                 FALSE
                      New value: TRUE
   UNWRAP:                       TRUE
                      New value: FALSE
   EXTRACTABLE:                  TRUE         (Cannot be changed from FALSE
                      New value: FALSE         to TRUE)
   SENSITIVE:                    FALSE        (Cannot be changed from TRUE
                      New value: TRUE          to FALSE)
```

Figure 142. ICSF Token Management - Private Key Object Details panel – Part 1

```
   ALWAYS SENSITIVE:             FALSE
   NEVER EXTRACTABLE:            FALSE
   FIPS140:                      FALSE
   WRAP WITH TRUSTED:            FALSE     (Cannot be changed from TRUE)
                     New value: TRUE
   IBM SECURE:                   FALSE
   IBM REGIONAL:                 FALSE
   APPLICATION:                  90893E31
   PRIVATE EXPONENT:             Not displayable
   PRIME 1:                      Not displayable
   PRIME 2:                      Not displayable
   EXPONENT 1:                   Not displayable
   EXPONENT 2:                   Not displayable
   COEFFICIENT:                  Not displayable
   PUBLIC EXPONENT:
      010001
   MODULUS:
      F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D
      18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC
      662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61
      5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805
      CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E
      69AD03FBFBBCC5990EDDF8C1A34D607AC3B7728D7E6ABBDA566A626980E0D888
      C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B
      95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562
      3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376
      CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF
      26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73
      C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0
      A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD
      7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0
      10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553
      1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753
Press ENTER to process.
Press END to exit to the previous menu.

COMMAND ===>
```

Figure 143. ICSF Token Management - Private Key Object Details panel – Part 2

If IBM SECURE is TRUE then panel will display these additional secure key attributes:

```
  IBM ALWAYS SECURE:         TRUE
  IBM ATTRBOUND:             FALSE
  IBM CARD COMPLIANCE:       FIPS2009
         New Value: Set to current processor value ___
                    FIPS 2009 _  BSI2009 _  (Or select one or
                    FIPS 2011 _  BSI 2011_   more from list)
```

With regional cryptographic keys, IBM SECURE is TRUE so the panel will display these additional secure key attributes:

```
  IBM ALWAYS SECURE:    TRUE or FALSE
  IBM ATTRBOUND:        FALSE
  IBM CARD COMPLIANCE:  NOT APPLICABLE
```

The format of the ICSF Token Management - Private Key Object Details panel will differ slightly depending on the type of key (RSA, DSA, Diffie-Hellman, Elliptic Curve, or SM2) selected.

Table 60. Information displayed in Private Key Object Details panel for RSA, DSA, Diffie-Hellman, Elliptic Curve, and SM2 keys

For this type of key:	Identified in the panel's KEY TYPE field as:	The panel will contain fields for:

RSA	RSA	Non-displayable private key information, the public key exponent, and the RSA modulus. For example:
		`PRIVATE EXPONENT:                    Not displayable`
		`PRIME 1:                             Not displayable`
		`PRIME 2:                             Not displayable`
		`EXPONENT 1:                          Not displayable`
		`EXPONENT 2:                          Not displayable`
		`COEFFICIENT:                         Not displayable`
		`PUBLIC EXPONENT:`
		`   010001`
		`MODULUS:`
		`   F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D`
		`   18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC`
		`   662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61`
		`   5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805`
		`   CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E`
		`   69AD03FBFBBCC5990EDDF8C1A34D607AC3B7728D7E6ABBDA566A626980E0D888`
		`   C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B`
		`   95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562`
		`   3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376`
		`   CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF`
		`   26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73`
		`   C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0`
		`   A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD`
		`   7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0`
		`   10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553`
		`   1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753`

DSA	DSA	The private key value (not displayable), the DSA prime *p*, subprime *q*, and base *g*. For example:
		`VALUE:                  Not displayable`
		`PRIME:`
		`   2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A`
		`   ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33`
		`   C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9`
		`   4490C4837B5656776E9FFDA073EAED869B19F7E197970DBE5665E8F87F964C57`
		`SUBPRIME:`
		`   DF9048626B9AD71EF6F33081890DF9048626B9AD`
		`BASE:`
		`   F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A`
		`   D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE`
		`   1362CDDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1`
		`   1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40`

Diffie-Hellman	DH	The private key value (not displayable), the size of the private key, and the Diffie-Hellman prime *p* and base *g*. For example:
		`VALUE:                  Not displayable`
		`VALUE BITS:             160`
		`PRIME:`
		`   F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D`
		`   18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC`
		`   662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61`
		`   5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805`
		`   CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E`
		`   69AD03FBFBBCC5990EDDF8C1A34D607AC3B7728D7E6ABBDA566A626980E0D888`
		`   C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B`
		`   95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562`
		`BASE:`
		`   3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376`
		`   CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF`
		`   26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73`
		`   C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0`
		`   A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD`
		`   7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0`
		`   10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553`
		`   1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753`

Table 60. Information displayed in Private Key Object Details panel for RSA, DSA, Diffie-Hellman, Elliptic Curve, and SM2 keys (continued)

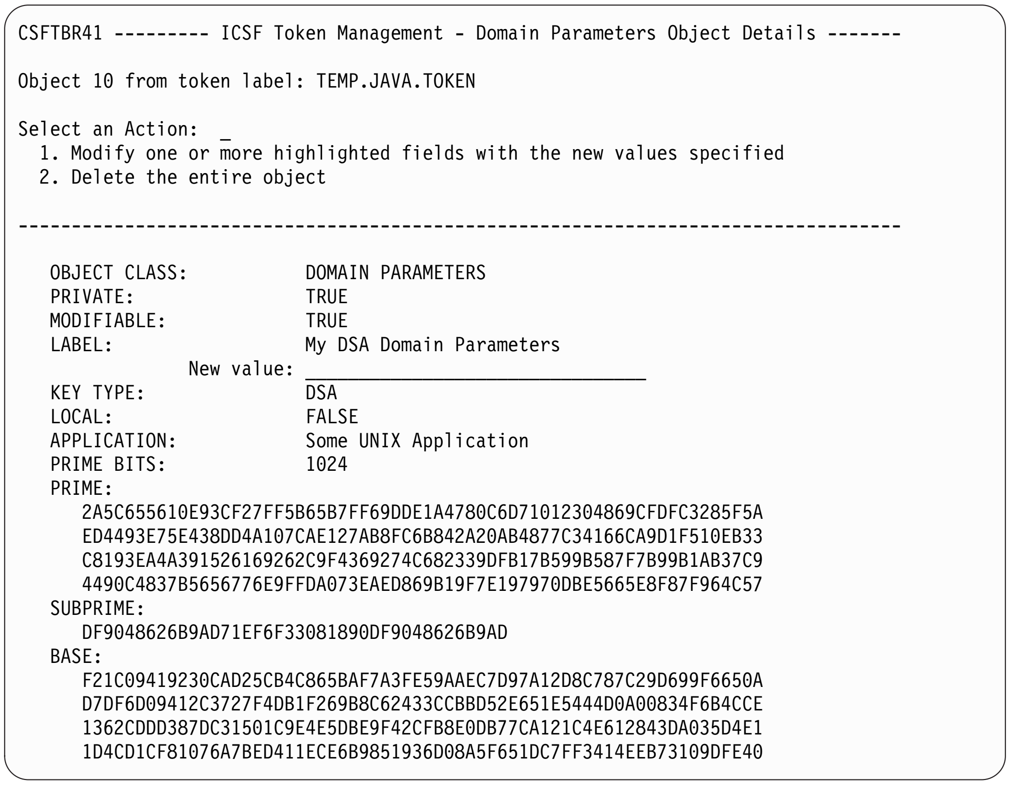
For this type of key:	Identified in the panel's KEY TYPE field as:	The panel will contain fields for:

Elliptic Curve	EC	The elliptic curve point. For example:
		`   VALUE:                  Not displayable`
		` _ EC PARAMS*:             Named Curve – secp521r1`

SM2	SM2	The elliptic curve parameters and the elliptic curve point. For example:
		`  VALUE:                  Not displayable`
		`_ EC PARAMS*:             Named Curve – GB-256`

Domain Parameters Object Details panel

If a domain parameter object is selected on the Token Details panel, the ICSF Token Management - Domain Parameters Object Details menu is presented:



```
CSFTBR41 --------- ICSF Token Management - Domain Parameters Object Details -------

Object 10 from token label: TEMP.JAVA.TOKEN

Select an Action:  _
  1. Modify one or more highlighted fields with the new values specified
  2. Delete the entire object

----------------------------------------------------------------------------------

   OBJECT CLASS:          DOMAIN PARAMETERS
   PRIVATE:               TRUE
   MODIFIABLE:            TRUE
   LABEL:                 My DSA Domain Parameters
                New value: ________________________________
   KEY TYPE:              DSA
   LOCAL:                 FALSE
   APPLICATION:           Some UNIX Application
   PRIME BITS:            1024
   PRIME:
      2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A
      ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33
      C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9
      4490C4837B5656776E9FFDA073EAED869B19F7E197970DBE5665E8F87F964C57
   SUBPRIME:
      DF9048626B9AD71EF6F33081890DF9048626B9AD
   BASE:
      F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A
      D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE
      1362CDDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1
      1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40
```

Figure 144. ICSF Token Management - Domain Parameters Object Details panel

The format of the ICSF Token Management - Domain Parameters Object Details panel will differ slightly depending on the domain parameter (DSA or Diffie-Hellman) selected.

Table 61. Information displayed in Domain Parameters Object Details panel for DSA and Diffie-Hellman domain parameters

For this type of key:	**Identified in the panel's KEY TYPE field as:**	**The panel will contain fields for:**

DSA	DSA	The DSA prime *p*, subprime *q*, and base *g*. For example:
		`KEY TYPE:              DSA`
		`LOCAL:                 FALSE`
		`APPLICATION:           Some UNIX Application`
		`PRIME BITS:            1024`
		`PRIME:`
		`   2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A`
		`   ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33`
		`   C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9`
		`   4490C4837B5656776E9FFDA073EAED869B19F7E197970DBE5665E8F87F964C57`
		`SUBPRIME:`
		`   DF9048626B9AD71EF6F33081890DF9048626B9AD`
		`BASE:`
		`   F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A`
		`   D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE`
		`   1362CDDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1`
		`   1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40`

Diffie-Hellman	DH	The Diffie-Hellman prime *p* and base *g*. For example:
		`KEY TYPE:              DH`
		`LOCAL:                 FALSE`
		`APPLICATION:           Some UNIX Application`
		`PRIME BITS:            2048`
		`PRIME:`
		`   F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D`
		`   18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC`
		`   662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61`
		`   5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805`
		`   CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E`
		`   69AD03FBFBBCC5990EDDF8C1A34D607AC3B7728D7E6ABBDA566A626980E0D888`
		`   C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B`
		`   95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562`
		`BASE:`
		`   3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376`
		`   CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF`
		`   26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73`
		`   C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0`
		`   A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD`
		`   7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0`
		`   10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553`
		`   1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753`

Chapter 17. Using the ICSF Utility Program CSFEUTIL

This topic contains Programming Interface Information.

ICSF provides a utility program, CSFEUTIL, that performs certain functions that can also be performed using the administrator's panels.

The program that executes CSFEUTIL must be APF-authorized.

The utility can be used for installations with cryptographic coprocessors. You can run the utility program to perform these tasks:
- Reencipher a disk copy of a CKDS
- Change the master key (AES or DES)
- Refresh the in-storage CKDS

You invoke the program as a batch job or from another program. To invoke the program as a batch job, use JCL. You specify different parameters on the EXEC statement depending on the task you want the utility program to perform. If the CSFEUTIL invocation from the batch job fails, you will need to invoke CSFEUTIL from another program to obtain the reason code from General Purpose Register 0 along with the return code in General Purpose Register 15. To invoke the program from another program, use standard MVS linkages like LINK, ATTACH, LOAD, and CALL.

Note: "CSFWEUTL" on page 348 provides sample code.

For information about using the utility program to reencipher a disk copy of a CKDS and change the master key, see "Symmetric Master Keys and the CKDS." For information about using the program to refresh the in-storage CKDS, see "Refreshing the in-storage CKDS using a utility program" on page 345.

Symmetric Master Keys and the CKDS

This topic describes how to use the utility program to reencipher a disk copy of a CKDS and to change a master key.

Note:
- Prior to performing any function that affects the current CKDS, such as reenciphering, refreshing, or changing the master key, consider temporarily disallowing dynamic CKDS update services. For more information, refer to "Steps for disallowing dynamic CKDS updates during CKDS administration updates" on page 186. If a CKDS reencipher is to be performed on a CKDS which is shared by members of a sysplex, dynamic CKDS updates should be disabled on all sysplex systems until the master key has been changed and the newly reenciphered CKDS is active on all systems sharing the CKDS
- If the CKDS contains HMAC keys, it must be reenciphered on a system with a CEX3C and the Sept. 2010 or later licensed internal code.

1. When you change a master key, you must first reencipher any disk copies of the CKDSs under the new master key in the new master key register.

   You can reencipher a CKDS either using the panels or the utility program.

Note:

- In compatibility or co-existence mode, you can use the utility program to reencipher a CKDS but not to change the master key. To change the master key using the utility program, you must be in noncompatibility mode.
- When invoking the master key reencipher you need access to the CSFMVR profile in the CSFSERV class.

2. Invoke the program as a batch job or from another program.

   You pass the same parameters whether you call the program as a batch job or from another program.
3. Pass the names of the CKDSs upon which to perform the task and the name of the task to perform.

   When you invoke the utility program from another program, General Register 1 must contain a pointer to the address of a data area whose structure is as follows:

   ```
   Bytes 0-1: Length of the parameter string in binary
   Bytes 2-n: The parameter string
   ```

   The parameter string is the same as that which you would specify using the PARM keyword on the EXEC JCL statement if you invoked the program as a batch job.
4. To reencipher a disk copy of a CKDS, pass these parameters in this order:
   a. The name of the disk copy of the CKDS to reencipher.
   b. The name of an empty disk copy of the CKDS to contain the reenciphered keys.
   c. The name for the task: REENC.

   Note: The input CKDS and the output CKDS must have the same VSAM attributes.
5. To reencipher the CKDS using JCL, use JCL like this example:

   ```
   //STEP EXEC PGM=CSFEUTIL,PARM='OLD.CKDS,NEW.CKDS,REENC'
   ```

   The first parameter passed, OLD.CKDS, is the name of the disk copy to reencipher. The second parameter, NEW.CKDS, is the name of an empty disk copy of the CKDS where you want ICSF to place the reenciphered keys.
6. When you reencipher all the disk copies of the CKDSs under the new master key, make the new master key active by changing the master key.

   The utility program activates the new master key and reads a disk copy of a CKDS reenciphered under the new master key into storage.
7. To change a master key, pass these parameters in this order:
   a. The name of the disk copy of the CKDS to read into storage.
   b. The name for the task: CHANGE.
8. To change the master key using JCL, use JCL like this example:

   ```
   //STEP EXEC PGM=CSFEUTIL,PARM='NEW.CKDS,CHANGE'
   ```

   The utility program reads the new master key into the master key register to make that master key active. The program also reads into storage a disk copy of the CKDS that you specify. This CKDS should be reenciphered under the new master key that you are making the current master key. The first parameter passed, NEW.CKDS, is the name of the disk copy of the CKDS that you want ICSF to read into storage.

When you invoke the program as a batch job, you receive the return code in a message when the job completes. You do not receive a reason code with the return code. When the program is invoked from another program, the invoking program receives the reason code in General Register 0 along with the return code in General Register 15. The return codes and reason codes are explained in "Return and reason codes for the CSFEUTIL program."

Refreshing the in-storage CKDS using a utility program

This topic describes how to use the CSFEUTIL program to refresh an in-storage CKDS.

1. Invoke the program from a batch job or from another program.
2. You pass the same parameters whether you call the program as a batch job or from another program.
3. Pass the names of the CKDSs to perform the task and the name for the task. When you invoke the utility program from another program, General Register 1 must contain a pointer to the address of a data area whose structure is as follows:
   ```
   Bytes 0-1: Length of the parameter string in binary
   Bytes 2-n: The parameter string
   ```
   The parameter string is the same as that which you would specify using the PARM keyword on the EXEC JCL statement if you invoked the program as a batch job.
4. To refresh an in-storage CKDS, pass these parameters in this order:
   - The name of the disk copy of the CKDS that you want read into storage
   - The name for the task: REFRESH
5. To refresh the CKDS using JCL, use JCL like this example:
   ```
   //STEP EXEC PGM=CSFEUTIL,PARM='NEW.CKDS,REFRESH'
   ```
   The first parameter passed, NEW.CKDS, is the name of the disk copy of the CKDS that you want read into storage.

Note: If a CKDS refresh is to be performed on a CKDS which is shared by members of a sysplex, dynamic CKDS updates should be disabled on all sysplex systems until the master key has been changed and the newly reenciphered CKDS is active on all systems sharing the CKDS

When you invoke the program as a batch job, you receive the return code in a message when the job completes. You do not receive a reason code with the return code. When the program is invoked from another program, the invoking program receives the reason code in General Register 0 along with the return code in General Register 15. The return codes and reason codes are explained in "Return and reason codes for the CSFEUTIL program."

Return and reason codes for the CSFEUTIL program

When you invoke the CSFEUTIL program as a batch job, you receive the return code in a message when the job completes. The meanings of the return codes are:

Return Code	Meaning
0	Process successful.
4	Parameters are incorrect.
8	RACF authorization check failed.
12	Process unsuccessful.

68 or 72
: CKDS processing has failed. An error was detected in the new KDS.

84 or 116
: CKDS processing encountered an abnormal end while processing a CKDS. When the return code is 84 the error occurred while processing the new CKDS. When the return code is 116 the error occurred while processing the old CKDS. View the SYSTRACE at the time of the error for an indication of the abend that was encountered.

100 or 104
: CKDS processing has failed. An error was detected in the old KDS.

101 or 105
: CKDS processing has failed. An error occurred while processing a KDS record. For a 101 return code, consult the Return Code 8 reason codes in the *z/OS Cryptographic Services ICSF Application Programmer's Guide*. For a 105 return code, consult the Return Code 12 reason codes in the *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

When the program is invoked from another program, the invoking program receives the reason code in General Register 0 along with the return code in General Register 15. The following list describes the meaning of the reason codes. If a particular reason code is not listed, refer to the listing of ICSF and TSS return and reason codes in the *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Return code 0 has these reason codes:

Reason Code	**Meaning**
36132	CKDS reencipher/Change MK processed only tokens encrypted under the DES master key.
36136	CKDS reencipher/Change MK processed only tokens encrypted under the AES master key.
36140	CKDS reencipher/Change MK processed tokens encrypted under the DES and AES master key.

Return code 8 has these reason codes:

Reason Code	**Meaning**
3114	Another refresh utility request is executing, and this utility request will not be allowed to run.
16000	Invoker has insufficient RACF access authority to perform function.

Return code 12 has these reason codes:

Reason Code	**Meaning**
36000	Unable to change master key. Check hardware status.
36020	Input CKDS is empty or not initialized (authentication pattern in the control record is invalid).
36060	The new master key register or registers are not full.

36068 The input KDS is not enciphered under the current master key.

36084 The master key register cannot be changed since ICSF is running in compatibility mode.

36104 Option not available. There were no Cryptographic Coprocessors available to perform the service that was attempted.

36160 The attempt to reencipher the CKDS failed because there is an enhanced token in the CKDS.

36168 The LRECL attribute of the input CKDS does not match the LRECL of the output CKDS.

36211 A request was made to load a key data set (CKDS, PKDS or TKDS)which has records which are in KDSR format. This level of ICSF does not support KDSR format records

Return code 72 or 104 has these reason codes:

Reason Code
Meaning

6008 A service routine has failed.

The service routines that may be called are:

CSFMGN
MAC generation

CSFMVR
MAC verification

CSFMKVR
Master key verification

6012 The Single-record, read-write installation exit (CSFSRRW) returned a return code greater than 4.

6016 An I/O error occurred reading or writing the CKDS.

6020 The CSFSRRW installation exit abended and the installation options EXIT keyword specifies that the invoking service should end.

6024 The CSFSRRW installation exit abended and the installation options EXIT keyword specifies that ICSF should end.

6028 The CKDS access routine could not establish the ESTAE environment.

6040 The CSFSRRW installation exit could not be loaded and is required.

6044 Information necessary to set up CSFSRRW installation exit processing could not be obtained.

6048 The system keys cannot be found while attempting to write a complete CKDS data set.

6052 For a write CKDS record request, the current master key verification pattern (MKVP) does not match the CKDS header record MKVP.

6056 The output CKDS is not empty.

36001 A variable-length record format CKDS cannot be used on a system with a Cryptographic Coprocessor Feature.

36168 The LRECL attribute of the input CKDS does not match the LRECL of the output CKDS.

Note: It is possible that you will receive MVS reason codes rather than ICSF reason codes, for example, if the reason code indicates a dynamic allocation failure. For an explanation of Dynamic Allocation reason codes, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

CSFWEUTL

CSFWEUTL invokes CSFEUTIL. CSFWEUTL is a sample program that contains sample JCL to assemble the sample program, sample link edit JCL to put the assembled sample program into an authorized library, and sample JCL that will invoke the sample program.

```
//<NAME>   JOB  <JOB CARD PARAMETERS>
//*****************************************************************
//*                                                               *
//*    Licensed Materials - Property of IBM                       *
//*    5650-ZOS                                                   *
//*    (C) Copyright IBM Corp. 2004, 2013                         *
//*                                                               *
//*                                                               *
//* This file contains a sample program (CSFWEUTL), sample JCL    *
//* to assemble the sample program, sample link edit JCL to put   *
//* the assembled sample program into an authorized library, and *
//* lastly sample JCL that will invoke the sample program.        *
//*                                                               *
//* CSFWEUTL: Invokes CSFEUTIL                                    *
//*                                                               *
//* DESCRIPTION:                                                  *
//* CSFEUTIL is an ICSF utility program that can perform certain  *
//* functions that can be performed by using the administrator's  *
//* panels.  The requested function is passed in the "PARM=..."   *
//* parameter.  Refer to the ICSF Administrator's Guide for       *
//* more information on CSFEUTIL functions.                       *
//*                                                               *
//* However, when running the ICSF CSFEUTIL, sometimes error      *
//* conditions may occur. The type of error is qualified by the   *
//* contents of register 15 and register 0 upon program exit.     *
//* Unfortunately, only register 15 (return code) is externalized *
//* when running these utilities from a batch JCL interface.      *
//*                                                               *
//* CSFWEUTL will call CSFEUTIL and pass any specified function in *
//* the "PARM=...  " parameter to CSFEUTIL.  On return from       *
//* CSFEUTIL, a WTO (write to operator) is issued containing      *
//* the return and reason codes.                                  *
//*                                                               *
//* CAUTION:                                                      *
//* This file contains four sample sections.  Before using this   *
//* sample, you have to make the following changes.               *
//*                                                               *
//* USER ACTIONS REQUIRED:                                        *
//* 1.Add the job parameters to meet your system requirements.    *
//*                                                               *
//* 2.In the ASSEMBLE JCL, change the SYSLIB DSN to match your    *
//*   installation specific data set names.                       *
//*                                                               *
//* 3.No changes are needed in the CSFWEUTL assembler code.       *
//*   This CSFWEUTL assembler code needs to reside in the         *
//*   SYSLIB DSN indicated in the ASSEMBLER JCL.                  *
//*                                                               *
//* 4.In the LKED JCL, for SYSLMOD DD statement, specify the      *
//*   installation specific authorized library dataset name that  *
//*   is to contain the CSFWEUTIL assembled code.                 *
//*                                                               *
//* 5.In the LKED JCL, for SYSLIB DD statement, specify your      *
//*   installation specific ICSF library dataset name.            *
//*   Change CSF to the appropriate high-level qualifier if you   *
//*   choose to not use the default.  If you use an edit or        *
//*   CHANGE command, be sure to include the period at the end      *
//*   of the high-level qualifier.                                  *
//*                                                                 *
//* 6.In the CSFWEUTL EXEC JCL, for the STEPLIB DSN, specify the    *
//*   same dataset name as was indicated in the SYSLMOD DSN         *
//*   statement in the LKED JCL.                                    *
//*                                                                 *
//* 7.In the CSFWEUTL EXEC JCL, for the PARM='....'  specify the    *
//*   requested function for CSFEUTIL.                              *
//*                                                                 *
//* 8.Users may want to separate the CSFWEUTL EXEC JCL into a       *
//*   separate JOB.                                                 *
//*                                                                 *
//* NOTES:                                                          *
//* 1.This job should be rerun with every new release of ICSF.      *
//*                                                                 *
//*******************************************************************
//*        JCL to assemble CSFWEUTL                                 *
//*******************************************************************
//* ASSEMBLER
//C        EXEC PGM=ASMA90,REGION=4M
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//         DD   DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD   DSN=&&SYSUT1,SPACE=(4096,(120,120),,,ROUND),UNIT=VIO,
//             DCB=BUFNO=1
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   DSN=&&LIN,DISP=(NEW,PASS),SPACE=(TRK,(2,2)),UNIT=SYSDA
//SYSIN    DD   *
*********************************************************************
*          CSFWEUTL assembler code                                  *
*********************************************************************

         TITLE 'CSFWEUTL   - ICSF CSFEUTIL INVOKER'
         PRINT GEN
***********************************************************************
*                                                                     *
* FUNCTION :  ICSF CSFEUTIL CALLER UTILITY                            *
*                                                                     *
* DESCRIPTIVE NAME : ICSF CSFEUTIL CALL ROUTINE                       *
*                                                                     *
* VERSION : RELEASE  1  LEVEL 000                                     *
*                                                                     *
* OBJECTIVE :                                                         *
*                                                                     *
* CSFEUTIL UTILITY :                                                  *
*                                                                     *
*   THIS PROGRAM ACCEPTS AN INVOCATION PARM THEN CALLS CSFEUTIL       *
*   PASSING THAT PARM. REGISTER 15 AND 0 ARE FORMATTED ON RETURN      *
*   IF NOT ZERO.  A WRITE TO OPERATOR IS THEN ISSUED.                 *
*                                                                     *
*                                                                     *
* DEPENDENCIES :                                                      *
*                                                                     *
*       1. UNDER OS/390 OPERATING SYSTEM                              *
*       2. UNDER IBM S/390                                            *
*       3. LANGUAGE : IBM S/390 ASSEMBLER                             *
*       4. ICSF UP AND ACTIVE                                         *
*                                                                     *
* ENTRY POINT :  CSFWEUTL                                             *
*                                                                     *
* INPUT ARGUMENTS : INVOCATION PARM PASSED TO CSFEUTIL                *
*                                                                     *
*                                                                     *
* OUTPUT ARGUMENTS :                                                  *
*                                                                     *
*     NONE                                                            *
*                                                                    *
* FUNCTION INPUT ARGUMENTS :                                         *
*                                                                    *
*     NONE                                                           *
*                                                                    *
* FUNCTION OUTPUT (RETURNS) :                                        *
*                                                                    *
*     RETCODE      R15SAVE                         (FULLWORD)        *
*                                                                    *
* EXIT-NORMAL RETURN CODE : 0                                        *
*                                                                    *
* EXIT-ERROR RETURN CODE : VALID RANGE 1 - 255                       *
*                                                                    *
* EXTERNAL-REFERENCES : NONE                                         *
*                                                                    *
* CHANGE ACTIVITY : NONE                                             *
*                                                                    *
**********************************************************************
R0       EQU   0
R1       EQU   1                       WORK REGISTER/CALL PARMS
R2       EQU   2                       WORK REGISTER
R3       EQU   3                       WORK REGISTER
R4       EQU   4                       WORK REGISTER
R5       EQU   5                       WORK REGISTER
R6       EQU   6                       WORK REGISTER
R7       EQU   7                       WORK REGISTER
R8       EQU   8                       WORK REGISTER
R9       EQU   9                       WORK REGISTER
R10      EQU   10                      WORK REGISTER
R11      EQU   11                      SECOND BASE REGISTER
R12      EQU   12                      BASE REGISTER
R13      EQU   13                      SAVE AREA CHAIN
R14      EQU   14                      RETURN ADDRESS
R15      EQU   15                      ENTRY POINT/RETURN CODE
         EJECT
CSFWEUTL CSECT
         USING CSFWEUTL,R12,R11        SET UP BASE REGISTER
         LA    R2,4095                       SET INCREMENT 4K
         LA    R2,1(R2)
         STM   R14,R12,12(R13)         SAVE REGISTERS
         LR    R12,R15                 SET UP ADDRESSABILITY
         LA    R11,0(R2,R12)                 SET SECOND BASE REG
         LA    R2,SAVEAREA
         ST    R13,4(R2)
         LR    R13,R2
         ST    R1,R1SAVE
         L     R4,0(R1)                GET INVOCATION PARM ADDRESS
         LH    R3,0(R4)                LOAD PARM LENGTH
         LTR   R3,R3                   ANY PARMS?
         BZ    NOPARM                  NO...BRANCH
         STH   R3,PARMLEN              SAVE PARM LENGTH
         BCTR  R3,0                    DECREMENT FOR EX
         LA    R4,2(R4)                POINT PAST LENGTH
         EX    R3,PARMSAVE             MOVE PARM TO INVOCATION FIELD
         B     START                   BRANCH AROUND CONSTANTS
         DC    C'** CSFWEUTL **'       MODULE
         DC    C'** &SYSDATE **'       ASM DATE
         DC    C'** &SYSTIME **'       ASM TIME
         DC C'CSFWEUTL : ICSF CSFEUTIL INVOCATION'
         DC C'    (C) COPYRIGHT IBM CORP. 2004 '
         DC C'LICENSED MATERIAL - PROGRAM PROPERTY OF IBM '
         EJECT
START    DS    0H
         OI    LINKPARM,X'80'          SET LAST PARM INDICATOR
         LA    R1,LINKPARM             LOAD PARM ADDRESS
         L     R15,=V(CSFEUTIL)        LOAD CSFEUTIL
         BALR  R14,R15                 INVOKE IT
         LTR   R15,R15                 ANY RETURN CODE?
         BZ    RETURN                  NO, ALL DONE
         ST    R0,R0SAVE               SAVE R0
         ST    R15,R15SAVE             SAVE R15
         L     R3,R15SAVE
         CVD   R3,DBWD                 DISPLAY R15 IN DECIMAL
         UNPK  UNPACK8(8),DBWD+4(4)
         OI    UNPACK8+7,X'F0'
         MVC   NOTZERO+23(8),UNPACK8
         L     R3,R0SAVE
         CVD   R3,DBWD                 DISPLAY R0 IN DECIMAL
         UNPK  UNPACK8(8),DBWD+4(4)
         OI    UNPACK8+7,X'F0'
         MVC   NOTZERO+37(8),UNPACK8
NOTZERO  WTO   'CSFWEUTL  R15: XXXXXXXX  R0: XXXXXXXX'
         B     RETURN
NOPARM   DS    0H
         WTO   'CSFWEUTL : NO PARAMETERS SPECIFIED'
         B     RETURN
RETURN   DS    0H
         L     R15,R15SAVE             GET CSFEUTIL RC
         L     R13,4(R13)
         ST    R15,16(13)
         LM    R14,R12,12(R13)
         BR    R14
         SPACE 3
PARMSAVE MVC   SAVEPARM(0),0(R4)
         SPACE 3
SAVEAREA DS    18F
R0SAVE   DS    F
R1SAVE   DS    F
R15SAVE  DS    F
DBWD     DS    D
UNPACK8  DS    D
         TITLE 'WORK AREAS'
         SPACE 3
         LTORG
         SPACE 3
LINKPARM DC    A(PARMLEN)
         DS    0D
PARMLEN  DC    H'0'
SAVEPARM DC    XL256'00'
         SPACE 3
         END   CSFWEUTL
//*******************************************************************
//*       JCL to link edit CSFWEUTL                                 *
//*******************************************************************
/*
//LKED     EXEC PGM=HEWL,PARM='MAP,LET,LIST,AC(1)',COND=(8,LT,C)
//SYSLIN   DD   DSN=&&LIN,DISP=(OLD,PASS)
//         DD   DDNAME=SYSIN
//SYSLMOD  DD   DSN=USER.STEPLIB,DISP=OLD
//SYSPRINT DD   SYSOUT=*
//SYSLIB   DD   DSN=CSF.SCSFMOD0,DISP=SHR
//*************************************************
//SYSIN    DD   *
 NAME CSFWEUTL(R)
//*******************************************************************
//*       JCL to invoke CSFWEUTL                                    *
//*******************************************************************
/*
//CSFWEUTL EXEC PGM=CSFWEUTL,REGION=512K,
//         PARM='CSF.EXAMPLE.CKDS,REFRESH'
//STEPLIB  DD   DSN=USER.STEPLIB,DISP=SHR
//*
```

Chapter 18. Using the ICSF Utility Program CSFPUTIL

This topic contains Programming Interface Information.

ICSF provides a utility program, CSFPUTIL, that performs certain functions that can also be performed using the administrator's panels.

You can run the utility program to perform these tasks:

- Reencipher a PKDS
- Refresh the in-storage copy of the PKDS

You invoke the program as a batch job or from another program. To invoke the program as a batch job, use JCL. You specify different parameters on the EXEC statement depending on the task you want the utility program to perform. To invoke the program from another program, use standard MVS linkages like LINK, ATTACH, LOAD, and CALL.

For information about using the utility program to reencipher a disk copy of a PKDS, see "Asymmetric master keys and the PKDS." For information about using the program to refresh the in-storage copy of the PKDS, see "Refreshing the in-storage copy of the PKDS" on page 354.

Asymmetric master keys and the PKDS

You can reencipher a PKDS either using the panels or the utility program.

1. Invoke the program as a batch job or from another program.

   You pass the same parameters whether you call the program as a batch job or from another program.
2. Pass the names of the PKDSs upon which to perform the task and the name of the task to perform.

   When you invoke the utility program from another program, General Register 1 must contain a pointer to the address of a data area whose structure is as follows:
   ```
   Bytes 0-1: Length of the parameter string in binary
   Bytes 2-n: The parameter string
   ```
   The parameter string is the same as that which you would specify using the PARM keyword on the EXEC JCL statement if you invoked the program as a batch job.
3. To reencipher a PKDS, pass these parameters in this order:
   a. The name of the PKDS to reencipher.
   b. The name of an empty PKDS to contain the reenciphered keys.
   c. The name for the task: RECIPHER.
4. To reencipher the PKDS using JCL, use JCL like this example:
   ```
   //STEP EXEC PGM=CSFPUTIL,PARM='OLD.PKDS,NEW.PKDS,RECIPHER'
   ```
   The first parameter passed, OLD.PKDS, is the name of the PKDS to reencipher. The second parameter, NEW.PKDS, is the name of an empty PKDS where you want ICSF to place the reenciphered keys.

5. When you reencipher all the PKDSs under the new master key, refresh the PKDS.

When you invoke the program as a batch job, you receive the return code in a message when the job completes. You do not receive a reason code with the return code. The return codes are explained in "Return and reason codes for the CSFPUTIL program."

Refreshing the in-storage copy of the PKDS

This topic describes how to use the CSFPUTIL program to refresh the in-storage copy of the PKDS.

1. Invoke the program from a batch job or from another program.

   You pass the same parameters whether you call the program as a batch job or from another program.
2. When you invoke the utility program from another program, General Register 1 must contain a pointer to the address of a data area whose structure is as follows:
   ```
   Bytes 0-1: Length of the parameter string in binary
   Bytes 2-n: The parameter string
   ```
   The parameter string is the same as that which you would specify using the PARM keyword on the EXEC JCL statement if you invoked the program as a batch job.
3. To refresh in-storage copy of the PKDS, pass this parameter:
   - The name for the task: REFRESH
   - Optional: the name of the disk copy of the PKDS you want read into storage. If no data set is specified, the active PKDS will be used.
4. To refresh the PKDS using JCL, use JCL like this example:
   ```
   //STEP EXEC PGM=CSFPUTIL,PARM='REFRESH,NEW.PKDS'
   ```
   The second parameter, NEW.PKDS, is the name of the disk copy of the PKDS that you want read into storage.
5. To refresh the active PKDS using JCL, use JCL like this example:
   ```
    //STEP EXEC PGM=CSFPUTIL,PARM='REFRESH'
   ```

When you invoke the program as a batch job, you receive the return code in a message when the job completes. You do not receive a reason code with the return code. The return codes are explained in "Return and reason codes for the CSFPUTIL program."

Return and reason codes for the CSFPUTIL program

When you invoke the CSFPUTIL program as a batch job, you receive the return code in a message when the job completes. The following list describes the meanings of the return codes. Additional return codes are described in "Return and reason codes for the CSFEUTIL program" on page 345.

Return Code	**Meaning**
0	Process successful.
2	Partially successful. Job completed but some tokens have not been reenciphered.

4 Parameters are incorrect. A possible cause of the error is that the parameter 'ACTIVATE' was used. That parameter is no longer supported; use 'REFRESH'.

8 RACF authorization failed.

12, 72, or 104
PKDS processing has failed. A return code 72 indicates the error was detected with the new KDS. A return code 104 indicates the error was detected with the old KDS.

When the program is invoked from another program, the invoking program receives the reason code in General Register 0 along with the return code in General Register 15. The meaning of the reason codes are as follows:

Return code 0 has these reason codes:

Reason Code	**Meaning**
36137	PKDS reencipher/Change MK processed only tokens encrypted under the RSA master key.
36138	PKDS reencipher/Change MK processed only tokens encrypted under the ECC master key.
36139	PKDS reencipher/Change MK processed tokens encrypted under the RSA and ECC master keys.

Return code 8 has this reason code:

Reason Code	**Meaning**
3114	Another refresh utility request is executing, and this utility request will not be allowed to run.
3080	Use of an unsupported token has been attempted. The usage of this type of token is not supported on the release of ICSF currently running.

Return code 12 has this reason code:

Reason Code	**Meaning**
36108	PKA callable services are enabled, and the PKDS is the active PKDS as specified in the options data set.
36116	PKDS specified for reencipher or activate has incorrect dataset attribute

An abend 18F Reason code x'300' occurs with a JCL error.

CSFWPUTL

CSFWPUTL invokes CSFPUTIL. CSFWPUTL is a sample program that contains sample JCL to assemble the sample program, sample link edit JCL to put the assembled sample program into an authorized library, and sample JCL that will invoke the sample program.

```
//<NAME>   JOB  <JOB CARD PARAMETERS>
//*****************************************************************
//*                                                               *
//*    Licensed Materials - Property of IBM                       *
//*    5650-ZOS                                                    *
//*    (C) Copyright IBM Corp. 2004, 2013                          *
//*                                                                *
//*                                                                *
//* This file contains a sample program (CSFWPUTL), sample JCL     *
//* to assemble the sample program, sample link edit JCL to put    *
//* the assembled sample program into an authorized library, and   *
//* lastly sample JCL that will invoke the sample program.         *
//*                                                                *
//* CSFWPUTL: Invokes CSFPUTIL                                     *
//*                                                                *
//* DESCRIPTION:                                                   *
//* CSFPUTIL is an ICSF utility program that can perform certain   *
//* functions that can be performed by using the administrator's   *
//* panels.  The requested function is passed in the "PARM=..."   *
//* parameter.  Refer to the ICSF Administrator's Guide for        *
//* more information on CSFPUTIL functions.                        *
//*                                                                *
//* However, when running the ICSF CSFPUTIL, sometimes error       *
//* conditions may occur. The type of error is qualified by the    *
//* contents of register 15 and register 0 upon program exit.      *
//* Unfortunately, only register 15 (return code) is externalized  *
//* when running these utilities from a batch JCL interface.       *
//*                                                                *
//* CSFWPUTL will call CSFPUTIL and pass any specified function in  *
//* the "PARM=...  " parameter to CSFPUTIL.  On return from        *
//* CSFPUTIL, a WTO (write to operator) is issued containing       *
//* the return and reason codes.                                   *
//*                                                                *
//* CAUTION:                                                       *
//* This file contains four sample sections.  Before using this    *
//* sample, you have to make the following changes.                *
//*                                                                *
//* USER ACTIONS REQUIRED:                                         *
//* 1.Add the job parameters to meet your system requirements.     *
//*                                                                *
//* 2.In the ASSEMBLE JCL, change the SYSLIB DSN to match your     *
//*   installation specific data set names.                        *
//*                                                                *
//* 3.No changes are needed in the CSFWPUTL assembler code.        *
//*   This CSFWPUTL assembler code needs to reside in the          *
//*   SYSLIB DSN indicated in the ASSEMBLER JCL.                   *
//*                                                                *
//* 4.In the LKED JCL, for SYSLMOD DD statement, specify the       *
//*   installation specific authorized library dataset name that   *
//*   is to contain the CSFWPUTIL assembled code.                  *
//*                                                                *
//* 5.In the LKED JCL, for SYSLIB DD statement, specify your       *
//*   installation specific ICSF library dataset name.             *
//*   Change CSF to the appropriate high-level qualifier if you    *
//*   choose to not use the default.  If you use an edit or        *
//*   CHANGE command, be sure to include the period at the end     *
//*   of the high-level qualifier.                                 *
//*                                                                *
//* 6.In the CSFWPUTL EXEC JCL, for the STEPLIB DSN, specify the   *
//*   same dataset name as was indicated in the SYSLMOD DSN        *
//*   statement in the LKED JCL.                                   *
//*                                                                *
//* 7.In the CSFWPUTL EXEC JCL, for the PARM='....'  specify the   *
//*   requested function for CSFPUTIL.                             *
//*                                                                *
//* 8.Users may want to separate the CSFWPUTL EXEC JCL into a      *
//*   separate JOB.                                                *
//*                                                                *
//* NOTES:                                                         *
//* 1.This job should be rerun with every new release of ICSF.     *
//*                                                                *
```

```
//*******************************************************************
//*        JCL to assemble CSFWPUTL                                 *
//*******************************************************************
//* ASSEMBLER
//C        EXEC PGM=ASMA90,REGION=4M
//SYSLIB   DD   DSN=SYS1.MACLIB,DISP=SHR
//         DD   DSN=SYS1.MODGEN,DISP=SHR
//SYSUT1   DD   DSN=&&SYSUT1,SPACE=(4096,(120,120),,,ROUND),UNIT=VIO,
//             DCB=BUFNO=1
//SYSPRINT DD   SYSOUT=*
//SYSLIN   DD   DSN=&&LIN,DISP=(NEW,PASS),SPACE=(TRK,(2,2)),UNIT=SYSDA
//SYSIN    DD   *
*********************************************************************
*          CSFWPUTL assembler code                                  *
*********************************************************************

         TITLE 'CSFWPUTL   - ICSF CSFPUTIL INVOKER'
         PRINT GEN
***********************************************************************
*                                                                     *
* FUNCTION :  ICSF CSFPUTIL CALLER UTILITY                            *
*                                                                     *
* DESCRIPTIVE NAME : ICSF CSFPUTIL CALL ROUTINE                        *
*                                                                     *
* VERSION : RELEASE  1  LEVEL 000                                     *
*                                                                     *
* OBJECTIVE :                                                         *
*                                                                     *
* CSFPUTIL UTILITY :                                                  *
*                                                                     *
*   THIS PROGRAM ACCEPTS AN INVOCATION PARM THEN CALLS CSFPUTIL       *
*   PASSING THAT PARM. REGISTER 15 AND 0 ARE FORMATTED ON RETURN      *
*   IF NOT ZERO.  A WRITE TO OPERATOR IS THEN ISSUED.                 *
*                                                                     *
*                                                                     *
* DEPENDENCIES :                                                      *
*                                                                     *
*       1. UNDER OS/390 OPERATING SYSTEM                              *
*       2. UNDER IBM S/390                                            *
*       3. LANGUAGE : IBM S/390 ASSEMBLER                             *
*       4. ICSF UP AND ACTIVE                                         *
*                                                                     *
* ENTRY POINT :  CSFWPUTL                                             *
*                                                                     *
* INPUT ARGUMENTS : INVOCATION PARM PASSED TO CSFPUTIL                 *
*                                                                     *
*                                                                     *
* OUTPUT ARGUMENTS :                                                  *
*                                                                     *
*     NONE                                                            *
*                                                                     *
* FUNCTION INPUT ARGUMENTS :                                          *
*                                                                     *
*     NONE                                                            *
*                                                                     *
* FUNCTION OUTPUT (RETURNS) :                                         *
*                                                                     *
*     RETCODE      R15SAVE                         (FULLWORD)         *
*                                                                     *
* EXIT-NORMAL RETURN CODE : 0                                         *
*                                                                     *
* EXIT-ERROR RETURN CODE : VALID RANGE 1 - 255                        *
*                                                                     *
* EXTERNAL-REFERENCES : NONE                                          *
*                                                                     *
* CHANGE ACTIVITY : NONE                                              *
*                                                                     *
```

```
***********************************************************************
R0       EQU   0
R1       EQU   1                        WORK REGISTER/CALL PARMS
R2       EQU   2                        WORK REGISTER
R3       EQU   3                        WORK REGISTER
R4       EQU   4                        WORK REGISTER
R5       EQU   5                        WORK REGISTER
R6       EQU   6                        WORK REGISTER
R7       EQU   7                        WORK REGISTER
R8       EQU   8                        WORK REGISTER
R9       EQU   9                        WORK REGISTER
R10      EQU   10                       WORK REGISTER
R11      EQU   11                       SECOND BASE REGISTER
R12      EQU   12                       BASE REGISTER
R13      EQU   13                       SAVE AREA CHAIN
R14      EQU   14                       RETURN ADDRESS
R15      EQU   15                       ENTRY POINT/RETURN CODE
         EJECT
CSFWPUTL CSECT
         USING CSFWPUTL,R12,R11         SET UP BASE REGISTER
         LA    R2,4095                        SET INCREMENT 4K
         LA    R2,1(R2)
         STM   R14,R12,12(R13)          SAVE REGISTERS
         LR    R12,R15                  SET UP ADDRESSABILITY
         LA    R11,0(R2,R12)                  SET SECOND BASE REG
         LA    R2,SAVEAREA
         ST    R13,4(R2)
         LR    R13,R2
         ST    R1,R1SAVE
         L     R4,0(R1)                 GET INVOCATION PARM ADDRESS
         LH    R3,0(R4)                 LOAD PARM LENGTH
         LTR   R3,R3                    ANY PARMS?
         BZ    NOPARM                   NO...BRANCH
         STH   R3,PARMLEN               SAVE PARM LENGTH
         BCTR  R3,0                     DECREMENT FOR EX
         LA    R4,2(R4)                 POINT PAST LENGTH
         EX    R3,PARMSAVE              MOVE PARM TO INVOCATION FIELD
         B     START                    BRANCH AROUND CONSTANTS
         DC    C'** CSFWPUTL **'        MODULE
         DC    C'** &SYSDATE **'        ASM DATE
         DC    C'** &SYSTIME **'        ASM TIME
         DC C'CSFWPUTL : ICSF CSFPUTIL INVOCATION'
         DC C'    (C) COPYRIGHT IBM CORP. 2004 '
         DC C'LICENSED MATERIAL - PROGRAM PROPERTY OF IBM '
         EJECT
START    DS    0H
         OI    LINKPARM,X'80'           SET LAST PARM INDICATOR
         LA    R1,LINKPARM              LOAD PARM ADDRESS
         L     R15,=V(CSFPUTIL)         LOAD CSFPUTIL
         BALR  R14,R15                  INVOKE IT
         LTR   R15,R15                  ANY RETURN CODE?
         BZ    RETURN                   NO, ALL DONE
         ST    R0,R0SAVE                SAVE R0
         ST    R15,R15SAVE              SAVE R15
         L     R3,R15SAVE
         CVD   R3,DBWD                  DISPLAY R15 IN DECIMAL
         UNPK  UNPACK8(8),DBWD+4(4)
         OI    UNPACK8+7,X'F0'
         MVC   NOTZERO+23(8),UNPACK8
         L     R3,R0SAVE
         CVD   R3,DBWD                  DISPLAY R0 IN DECIMAL
         UNPK  UNPACK8(8),DBWD+4(4)
         OI    UNPACK8+7,X'F0'
         MVC   NOTZERO+37(8),UNPACK8
NOTZERO  WTO   'CSFWPUTL  R15: XXXXXXXX  R0: XXXXXXXX'
         B     RETURN
NOPARM   DS    0H
```

```
         WTO   'CSFWPUTL : NO PARAMETERS SPECIFIED'
         B     RETURN
RETURN   DS    0H
         L     R15,R15SAVE            GET CSFPUTIL RC
         L     R13,4(R13)
         ST    R15,16(13)
         LM    R14,R12,12(R13)
         BR    R14
         SPACE 3
PARMSAVE MVC   SAVEPARM(0),0(R4)
         SPACE 3
SAVEAREA DS    18F
R0SAVE   DS    F
R1SAVE   DS    F
R15SAVE  DS    F
DBWD     DS    D
UNPACK8  DS    D
         TITLE 'WORK AREAS'
         SPACE 3
         LTORG
         SPACE 3
LINKPARM DC    A(PARMLEN)
         DS    0D
PARMLEN  DC    H'0'
SAVEPARM DC    XL256'00'
         SPACE 3
         END   CSFWPUTL
//*******************************************************************
//*       JCL to link edit CSFWPUTL                                 *
//*******************************************************************
/*
//LKED     EXEC PGM=HEWL,PARM='MAP,LET,LIST,AC(1)',COND=(8,LT,C)
//SYSLIN   DD   DSN=&&LIN,DISP=(OLD,PASS)
//         DD   DDNAME=SYSIN
//SYSLMOD  DD   DSN=USER.STEPLIB,DISP=OLD
//SYSPRINT DD   SYSOUT=*
//SYSLIB   DD   DSN=CSF.SCSFMOD0,DISP=SHR
//*****************************************************
//SYSIN    DD   *
 NAME CSFWPUTL(R)
//*******************************************************************
//*       JCL to invoke CSFWPUTL                                    *
//*******************************************************************
/*
//CSFWPUTL EXEC PGM=CSFWPUTL,REGION=512K,
//         PARM='CSF.EXAMPLE.PKDS,REFRESH'
//STEPLIB  DD   DSN=USER.STEPLIB,DISP=SHR
//*
```

Chapter 19. Using the ICSF Utility Program CSFDUTIL

ICSF provides a utility program, CSFDUTIL, that reads through a CKDS or PKDS and generates a report for duplicate key tokens.

Using the Duplicate Token Utility

There is no panel interface to this utility. The key data set must be specified as either a CKDS or a PKDS.

1. Invoke the program as a batch job
2. You must have UPDATE authority to the profile in the DATASET class covering the KDS.

   To generate a report for a CKDS with the fully qualified data set name of ICSF.HCR7751.CKDS, use this JCL example:

   ```
   //DUTIL     EXEC PGM=CSFDUTIL
   //SYSOUT    DD SYSOUT=*
   //SYSIN     DD *
     CKDSN(ICSF.HCR7751.CKDS)
   /*
   //
   ```

The supported option is either:

```
CKDSN(fully-qualified-CKDS-name)

     or

PKDSN(fully-qualified-PKDS-name)
```

When you invoke the program as a batch job, you receive the return and reason code in a message when the job completes. The return codes are explained in “Return and reason codes for the CSFDUTIL program” on page 362.

The data set name is assumed to be fully-qualified.

Note: Prior to analyzing the current CKDS or PKDS, consider temporarily disallowing dynamic CKDS and PKDS update services. For more information, refer to "Steps for disallowing dynamic CKDS updates during KGUP updates". If the analysis is to be performed on a CKDS or PKDS which is shared by members of a sysplex, dynamic updates of the CKDS and PKDS should be disabled on all sysplex systems until the analysis job is complete.

CSFDUTIL output

The CKDS information that is written out has the format:

Table 62. CKDS information from CSFDUTIL

Column	Value
1 - 64	Key label
67 - 74	Key type from the KDS record
77 - 84	Creation date. yyyymmdd
87 - 94	Creation time. hhmmssth
97 - 104	Last update date. yyyymmdd

Table 62. CKDS information from CSFDUTIL (continued)

Column	Value
107 - 114	Last update time. hhmmssth

The PKDS information that is written out has the format:

Table 63. PKDS information from CSFDUTIL

Column	Value
1 - 64	Key label
67 - 74	Creation date. yyyymmdd
77 - 84	Creation time. hhmmssth
87 - 94	Last update date. yyyymmdd
97 - 104	Last update time. hhmmssth

Return and reason codes for the CSFDUTIL program

When you invoke the CSFDUTIL program as a batch job, you receive the return code in a message when the job completes. The meanings of the return codes are:

Return Code	Meaning
0	Processing completed successful.
4	Parameters are incorrect.
8	RACF authorization check failed.
12	Processing unsuccessful. Additional messages issued.
16	Processing unsuccessful. Additionally, there was an error issuing diagnostic messages.
20	An ABEND occurred.

When the program is invoked from another program, the invoking program receives the reason code in General Register 0 along with the return code in General Register 15. The meaning of the reason codes are as follows:

Return code 0 has this reason code:

Reason Code	Meaning
0	Processing completed successfully.

Return code 4 has this reason code:

Reason Code	Meaning
32	There was an error in the options provided. See the output for details.

Return code 8 has this reason code:

Reason Code
Meaning

1600 Invoker has insufficient RACF access authority to use this service.

Return code 12 has these reason codes:

Reason Code
Meaning

6016 An IO error has occurred. See the output for details.

6028 There was an error establishing an ESTAE.

6032 There was a error allocating a data set or DD. See the output for details.

6036 There was an error deallocating a data set or DD. See the output for details.

36211 A request was made to load a key data set (CKDS, PKDS or TKDS)which has records which are in KDSR format. This level of ICSF does not support KDSR format records

Return code 16 has the same reason code as return code 12, but indicates that an error occurred in writing to the output in addition to the initial error.

Return code 20 has this reason code:

Reason Code
Meaning

4 An abnormal ending occurred. Contact your system programmer or the IBM Support Center.

CSFWDUTL

CSFWDUTL is a batch job that invokes the duplicate utilities program (CSFDUTIL). It can be found in SYS1.SAMPLIB.

```
//<NAME>   JOB  <JOB CARD PARAMETERS>
//*****************************************************************
//*                                                               *
//*    Licensed Materials - Property of IBM                       *
//*    5650-ZOS                                                   *
//*    Copyright IBM Corp. 2008, 2013                             *
//*                                                               *
//*                                                               *
//* This file contains sample JCL to invoke the Duplicate Token   *
//* Utility program (CSFDUTIL).                                   *
//*                                                               *
//* CSFWDUTL: Invokes CSFDUTIL                                    *
//*                                                               *
//* DESCRIPTION:                                                  *
//* CSFDUTIL is an ICSF utility program that searches the CKDS or *
//* PKDS for duplicate key tokens.  This utility is not available *
//* from the ICSF PANELS.  Refer to the ICSF Administrator's      *
//* Guide for more information on CSFDUTIL functions.             *
//*                                                               *
//* CSFWDUTL calls CSFDUTIL and passes the requested function into *
//* the utility using the SYSIN dd statement.                     *
//*                                                               *
//* CAUTION:                                                      *
//* Before using this sample, you have to make the following      *
//* changes.                                                      *
//*                                                               *
//* USER ACTIONS REQUIRED:                                        *
//* 1.Add the job parameters to meet your system requirements.    *
//* 2.If necessary change the parameter value to the key data set *
//*   you want to examine.                                        *
//*                                                                    *
//**********************************************************************
//*       JCL to invoke CSFDUTIL                                       *
//**********************************************************************
//CSFWDUTL EXEC PGM=CSFDUTL
//SYSOUT    DD  SYSOUT=*
//SYSIN     DD  *
  CKDSN(CSF.CSFCKDS)
/*
```

Chapter 20. Rewrapping DES key token values in the CKDS using the utility program CSFCNV2

ICSF provides a utility program, CSFCNV2, that will rewrap all encrypted DES tokens in the CKDS.

Note: You can also use the CSFCNV2 utility to convert a fixed-length record format CKDS to a variable-length record format. For more information on this capability of the CSFCNV2 utility, refer to *z/OS Cryptographic Services ICSF System Programmer's Guide*.

As described in "DES key wrapping" on page 25, there are two methods for wrapping the key value in a DES key token. The original method encrypts DES tokens using triple DES encryption. An enhanced wrapping method, introduced in FMID HCR7780 and designed to be ANSI X9.24 compliant, bundles the keys with other token data and encrypts the keys and associated data using triple DES encryption.

Using the CSFCNV2 utility, you can rewrap all encrypted key tokens in the CKDS using the enhanced or the original method. The results will be written to a new CKDS.

There is no panel interface for this utility. It can be invoked as a batch job and requires a z196 or new server.

To rewrap encrypted key tokens in an existing CKDS and write the results to a new CKDS, use the following JCL code as an example:

```
//STEP EXEC PGM=CSFCNV2,PARM='WRAP-xxx,OLD.CKDS,NEW.CKDS'
```

Where:

WRAP-*xxx*
: Specifies the wrapping method to use.

  WRAP-ECB
  : The original wrapping method. If you specify this option, be aware that the access control point "CKDS Conversion2 utility - Convert from enhanced to original" must be enabled. This access control point is not enabled in the ICSF coprocessor role. It can only be enabled using TKE.

  WRAP-ENH
  : The enhanced wrapping method.

  ENH-ONLY
  : The enhanced wrapping method will be used and the control vector in tokens will be updated to indicate that token cannot be rewrapped to the original method.

OLD.CKDS
: The name of the disk copy of the CKDS to process.

NEW.CKDS
: The name of an empty disk copy of the CKDS to contain the rewrapped keys.

The CSFV0560 message in the joblog will indicate the results of processing.

Return Code	Meaning
0	Process successful.
4	Minor error occurred.
8	RACF authorization check failed.
12	Process unsuccessful.
60 or 92	CKDS processing has failed. A return code 60 indicates the error was detected in the new KDS. A return code 92 indicates the error was detected with the old KDS.

When the program is invoked from another program, the invoking program receives the reason code in General Register 0 along with the return code in General Register 15. The following list describes the meaning of the reason codes. If a particular reason code is not listed, refer to the listing of ICSF and TSS return and reason codes in the *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Return code 0 has this reason code:

Reason Code	Meaning
36132	CKDS reencipher/Change MK processed only tokens encrypted under the DES master key.

Return code 4 has these reason codes:

Reason Code	Meaning
0	Parameters are incorrect.
4004	Rewrapping is not allowed for one or more keys.
36112	CKDS conversion completed successfully but some tokens could not be rewrapped because the control vector prohibited rewrapping from the enhanced wrapping method.
36164	Input CKDS is already in the variable-length record format. No conversion is necessary.

Return code 8 has this reason code:

Reason Code	Meaning
16000	Invoker has insufficient RACF access authority to perform function.

Return code 12 has these reason codes:

Reason Code	Meaning
0	ICSF has not been started
11060	The required cryptographic coprocessor was not active or the master key has not been set

36020 Input CKDS is empty or not initialized (authentication pattern in the control record is invalid).

36068 The input KDS is not enciphered under the current master key.

36104 Option not available. There were no Cryptographic Coprocessors available to perform the service that was attempted.

36160 The attempt to reencipher the CKDS failed because there is an enhanced token in the CKDS.

36168 A CKDS has an invalid LRECL value for the requested function. For wrapping, the input and output CKDS LRECLs must be the same.

36172 The level of hardware required to perform the operation is not available.

Return code 60 or 92 has these reason codes:

Reason Code
Meaning

3078 The CKDS was created with an unsupported LRECL.

5896 The CKDS does not exist.

6008 A service routine has failed.

The service routines that may be called are:

CSFMGN
MAC generation

CSFMVR
MAC verification

CSFMKVR
Master key verification

6012 The Single-record, read-write installation exit (CSFSRRW) returned a return code greater than 4.

6016 An I/O error occurred reading or writing the CKDS.

6020 The CSFSRRW installation exit abended and the installation options EXIT keyword specifies that the invoking service should end.

6024 The CSFSRRW installation exit abended and the installation options EXIT keyword specifies that ICSF should end.

6028 The CKDS access routine could not establish the ESTAE environment.

6040 The CSFSRRW installation exit could not be loaded and is required.

6044 Information necessary to set up CSFSRRW installation exit processing could not be obtained.

6048 The system keys cannot be found while attempting to write a complete CKDS data set.

6052 For a write CKDS record request, the current master key verification pattern (MKVP) does not match the CKDS header record MKVP.

6056 The output CKDS is not empty.

Note: It is possible that you will receive MVS reason codes rather than ICSF reason codes, for example, if the reason code indicates a dynamic allocation failure.

For an explanation of Dynamic Allocation reason codes, see *z/OS MVS Programming: Authorized Assembler Services Guide*

Chapter 21. Using ICSF health checks

The IBM Health Checker for z/OS is used to identify potential problems before they impact availability or cause outages. The Health Checker outputs messages to notify the user of the problems and suggests actions to be taken. The messages can be merely informational, or they can indicate a risk to the operation of the product.

ICSF provides a set of health checks to inform the user of potential ICSF problems. The checks include both migration checks and status checks. A migration check is designed to warn of changes in a current or pending ICSF release that could negatively impact usage. A status check provides information on the current state of ICSF.

The ICSF health checks are (see the ICSF health check's description for availability):
- ICSF_COPROCESSOR_STATE_NEGCHANGE
- ICSF_DEPRECATED_SERV_WARNINGS
- ICSF_KEY_EXPIRATION
- ICSF_MASTER_KEY_CONSISTENCY
- ICSF_OPTIONS_CHECKS
- ICSF_UNSUPPORTED_CCA_KEYS

The ICSF migration checks are (see the ICSF migration check's description for availability):
- ICSFMIG_DEPRECATED_SERV_WARNINGS
- ICSFMIG_MASTER_KEY_CONSISTENCY
- ICSFMIG7731_ICSF_RETAINED_RSAKEY
- ICSFMIG77A1_COPROCESSOR_ACTIVE
- ICSFMIG77A1_TKDS_OBJECT
- ICSFMIG77A1_UNSUPPORTED_HW

RACF provides a set of health checks that examine the status of general resource classes and also the security characteristics of system-critical data sets. The ICSF-related RACF health checks are:
- RACF_SENSITIVE_RESOURCES, for the ICSF key data sets.
- RACF_CSFSERV_ACTIVE, for the CSFSERV resource class.
- RACF_CSFKEYS_ACTIVE, for the CSFSERV resource class.

See *IBM Health Checker for z/OS User's Guide* for more information.

SAF Authorization for ICSF health checks

Some of the ICSF health checks use callable services to gather information for the check. If ICSF is running with CHECKAUTH(YES) specified in the options data set and the access to service through the CSFSERV SAF class is fully controlled, authority to specific services is required for some health checks.

The following health checks require access to the listed CSFSERV service profiles:
- ICSF_KEY_EXPIRATION
  - CSFKDSL

  - CSFKDMR
- ICSFMIG7731_ICSF_RETAINED_RSAKEY
  - CSFRKL
- ICSF_UNSUPPORTED_CCA_KEYS
  - CSFKDSL

Accessing the ICSF health checks

Health checks can be accessed using the System Display and Search Facility (SDSF) option in ISPF. SDSF provides a CK option to access the Health Checker:

```
HQX7780 -----------------  SDSF PRIMARY OPTION MENU  ------------------------

.DA    Active users                     .INIT  Initiators
.I     Input queue                      .PR    Printers
.O     Output queue                     .PUN   Punches
.H     Held output queue                .RDR   Readers
.ST    Status of jobs                   .LINE  Lines
                                        .NODE  Nodes
.LOG   System log                       .SO    Spool offload
.SR    System requests                  .SP    Spool volumes
.MAS   Members in the MAS               .NS    Network servers
.JC    Job classes                      .NC    Network connections
.SE    Scheduling environments
.RES   WLM resources                    .RM    Resource monitor
.ENC   Enclaves                         .CK    Health checker
.PS    Processes
                                        .ULOG  User session log
.END   Exit SDSF
```

Selecting the Health Checker (CK) option displays the available checks. The checks are displayed alphabetically by name. The ICSF checks start with 'ICSF'.

```
SDSF HEALTH CHECKER DISPLAY  SY1
PREFIX=*  DEST=(ALL)  OWNER=*  SYSNAME=
NP .NAME                            .CheckOwner  .State            .Status
     GRS_EXIT_PERFORMANCE             IBMGRS      ACTIVE(ENABLED)   SUCCESSFUL
     GRS_GRSQ_SETTING                 IBMGRS      ACTIVE(DISABLED)  ENV N/A
     GRS_MODE                         IBMGRS      ACTIVE(DISABLED)  ENV N/A
     GRS_RNL_IGNORED_CONV             IBMGRS      ACTIVE(DISABLED)  ENV N/A
     GRS_SYNCHRES                     IBMGRS      ACTIVE(ENABLED)   SUCCESSFUL
     ICSF_COPROCESSOR_STATE_NEGCHANGE IBMICSF     ACTIVE(ENABLED)   SUCCESSFUL
     ICSF_KEY_EXPIRATION              IBMICSF     ACTIVE(ENABLED)   SUCCESSFUL
     ICSF_MASTER_KEY_CONSISTENCY      IBMICSF     ACTIVE(ENABLED)   SUCCESSFUL
     ICSF_DEPRECATED_SERV_WARNINGS    IBMICSF     INACTIVE(ENABLED) INACTIVE
     ICSFMIG7731_ICSF_RETAINED_RSAKEY IBMICSF     INACTIVE(ENABLED) INACTIVE
     IEA_ASIDS                        IBMSUP      ACTIVE(ENABLED)   SUCCESSFUL
     IEA_LXS                          IBMSUP      ACTIVE(ENABLED)   SUCCESSFUL
     IOS_CAPTUCB_PROTECT              IBMIOS      ACTIVE(ENABLED)   SUCCESSFUL
     IOS_CMRTIME_MONITOR              IBMIOS      ACTIVE(ENABLED)   SUCCESSFUL
     IOS_MIDAW                        IBMIOS      ACTIVE(ENABLED)   SUCCESSFUL
     IOS_STORAGE_IOSBLKS              IBMIOS      ACTIVE(ENABLED)   SUCCESSFUL
```

The coprocessor state degradation check is enabled when ICSF is started and will monitor the coprocessor states on a daily basis until deactivated. The master key consistency check is enabled when ICSF is started and monitors the consistency of the states of each master key across the active and online coprocessors. The two migration checks are inactive when ICSF is started and must be activated to perform their checks.

ICSF_COPROCESSOR_STATE_NEGCHANGE

Type: Status

Initial State: Active

Interval: Daily

z/OS and ICSF releases the check applies to: ICSF FMID HCR7790 and later running on z/OS V1R12 and later.

This is a status check. The check detects a degradation in the state of any cryptographic coprocessor or accelerator on the system. The check is activated during the initialization of ICSF. The check is performed on a daily basis.

A state degradation is reported by AP number for the cryptographic coprocessor or accelerator. The states are described in the "Displaying cryptographic coprocessor status" on page 267. A state degradation has a possible negative impact on the operation of ICSF and the dependent cryptographic workload. The cause of the change should be understood.

The check output is obtained by selecting (s) on the Health Checker menu:

```
CHECK(IBMICSF,ICSF_COPROCESSOR_STATE_NEGCHANGE)
START TIME: 05/23/2011 14:33:49.364933
CHECK DATE: 20110320  CHECK SEVERITY: MEDIUM

* Medium Severity Exception *

CSFH0010E Coprocessor or Accelerator with AP number 35
has changed from ACTIVE state to OFFLINE state.

  Explanation:  The Coprocessor or accelerator state has degraded since
    the last check.

  System Action:  This has a possible negative impact on the operation
    of ICSF and the dependent cryptographic workload.

  Operator Response:  Report this exception to the System Programmer.

  System Programmer Response:  Alert the installation security
    Administrator to determine the impact of the change in coprocessor
    state.

  Problem Determination:  Refer to the ICSF Coprocessor Management and
    hardware status panels and the support element (SE) panel for
    further information regarding the coprocessors.

  Source:  Integrated Cryptographic Service Facility (ICSF)

  Reference Documentation:  z/OS Cryptographic Services Integrated
    Cryptographic Service Facility: Systems Programmers Guide.

  Automation:  n/a

  Check Reason:  Detects degradation in coprocessor state.

END TIME: 05/23/2011 14:37:36.608096  STATUS: EXCEPTION-MED
```

ICSF_DEPRECATED_SERV_WARNINGS

Type: Status

Initial State: Active

Interval: Daily

z/OS and ICSF releases the check applies to: ICSF FMID HCR77B0 and later running on z/OS V1R13 and later.

This is a deprecated services health check. The check detects the use of services that are no longer enhanced and are not recommended for continued use. The check is activated when the ICSF task is started and runs on a periodic (daily) basis.

The check output is obtained by selecting (s) the check on the Health Checker menu. If the check determines that deprecated services are being used then an exception is generated.

The check output is obtained by selecting (s) on the Health Checker menu:

```
CHECK(IBMICSF,ICSF_DEPRECATED_SERV_WARNINGS)
START TIME: 05/20/2014 08:33:39.248906
CHECK DATE: 20140520  CHECK SEVERITY: LOW

* Low Severity Exception *

  CSFH0011I Cryptographic Service CSFEDC is currently used, but this
  service has been deprecated.

  Explanation: The specified callable service is no longer being enhanced
    and is not recommended for continued use. This service may be removed
    in future releases, thus in the future, workloads using the service may
    fail.

  System Action:  There is no effect on this system.

  Operator Response:  Report this exception to the System Programmer.

  System Programmer Response:  Alert the installation security
    Administrator and application/middleware administrators for this
    system.

  Problem Determination:  Investigate applications using this service
    and determine appropriate actions to remove or replace the use of
    this service.

  Source:  Integrated Cryptographic Service Facility (ICSF)

  Reference Documentation:  z/OS Cryptographic Services Integrated
    Cryptographic Service Facility: Application Programmers Guide
    (ICSF FMID HCR77A1 and later).

  Automation:  n/a

  Check Reason:  Detects use of deprecated callable service.

END TIME: 05/20/2014 08:35:40.308973  STATUS: EXCEPTION-LOW
```

The deprecated services checked in this release are:

- CSFEDC

- CSFEMK
- CSFGKC
- CSFRTC

ICSF_KEY_EXPIRATION

Type: Status

Initial State: Active

Interval: Daily

z/OS and ICSF releases the check applies to: ICSF FMID HCR77B0 and later running on z/OS V1R13 and later.

This is a status check. The check detects records in the active key data sets that have the key material validity end date metadata set and will expire within the specified interval. The active CKDS, PKDS, and TKDS are checked. The label of all records that will expire will be listed along with the expiration date.

Note: The key data sets must use the KDSR format (introduced in ICSF FMID HCR77A1) in order to have key material validity dates. For additional details, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

The interval is set by the DAYS(*nnn*) parameter. The default interval is 60 days.

The check is activated during the initialization of ICSF. The check is performed on a daily basis.

When the ICSF_KEY_EXPIRATION health check is run, the following messages are generated:
- Message CSFH0030I is an informational message that displays the health check header.
- Message CSFH0032I indicates that there are no records that are about to expire.
- Message CSFH0031E indicates that there are records that are about to expire.

For example:

```
CHECK(ICSF,ICSF_KEY_EXPIRATION)
START TIME: 03/23/2015 08:10:01.603497
CHECK DATE: 20150101 CHECK SEVERITY: MEDIUM

* Medium Severity Exception *

CSFH0030I Cryptographic Keys Expiring in 60 Days
Active CKDS: CSF.CKDS

Records expiring on 20150401
CSF.SPECIAL.KEY.FOR.TESTING.ABCD0001              EXPORTER
CSF.SPECIAL.KEY.FOR.TESTING.ABCD0004              IMPORTER

Records expiring on 20150430
CSF.SPECIAL.KEY.FOR.TESTING.ABCD0002              MAC

Active PKDS: CSF.PKDS
Key data set not in KDSR format

CSFH0032E Check detected KDS record that will expire within the next 60 days.
```

```
Explanation: This check detected keys in the key data sets that will reach their
expiration date within the specified interval. When the keys reach their expiration
date, the keys can no longer be used the applications.

System action: There is no effect on the system.

Operator response: Contact the ICSF administrator.

System Programmer Response: Contact the ICSF administrator.

Problem Determination: n/a

Source: n/a

Reference Documentation: z/OS Cryptographic Services
Integrated Cryptographic Service Facility: Administrator's
Guide

Automation: n/a

Check Reason: Detects operational keys that will expire
within the specified interval.

END TIME: 03/23/2015 08:10:01.643285 STATUS: SUCCESSFUL

Active TKDS: CSF.TKDS

Objects expiring on 20150401
CSF.SPECIAL.TOKEN.FOR.TEST.AD0  0000000AY

Objects expiring on 20150421
CSF.SPECIAL.TOKEN.FOR.TEST.AD0  0000001AY

Objects expiring on 20150521
CSF.SPECIAL.TOKEN.FOR.TEST.AD0  0000011AY

CSFH0033E Check detected KDS record that will expire within
the next 60 days.

Explanation: This check detected keys in the key data sets
that will reach their expiration date within the specified
interval When the keys reach their expiration date, the
keys can no longer be used the applications.

System action: There is no effect on the system.

Operator response: Contact the ICSF administrator.

System Programmer Response: Contact the ICSF administrator.

Problem Determination: n/a

Source: n/a

Reference Documentation: z/OS Cryptographic Services
Integrated Cryptographic Service Facility:
Administrator's Guide

Automation: n/a

Check Reason: Detects operational keys that will expire
within the specified interval.

END TIME: 03/23/2015 08:10:01.643285 STATUS: SUCCESSFUL
```

ICSF_MASTER_KEY_CONSISTENCY

Type: Status

Initial State: Active

Interval: Daily

z/OS and ICSF releases the check applies to: ICSF FMID HCR77A0 and later running on z/OS V1R12 and later.

This is a master key health check. The check detects inconsistencies in the states of the coprocessor master keys. The check is activated when the ICSF task is started and runs on a periodic (daily) basis. The check determines when the state of a master key on at least one coprocessor is not in accord with the state on the other coprocessors.

The master key states for the coprocessors are displayed on the ICSF Coprocessor Management panel. The states can be available ("A"), correct ("C"), error ("E"), uninitialized ("U") or not supported (-). Available indicates that the master key loaded on the Coprocessor matches the master key used in the CKDS/PKDS/TKDS and is available for use. Correct indicates that the key matches the key used in the CKDS/PKDS/TKDS but is not available for use. Error indicates that the key does not match the key used in the CKDS/PKDS/TKDS. Uninitialized indicates that the key has not been set. Master keys are identified by Master Key Verification Pattern (MKVP).

The check is instituted to assist the user in maintaining master key functionality. The coprocessor activation algorithm maximizes the number of active cryptographic coprocessors. For non-CCF systems any valid master key is acceptable for coprocessor activation. To activate the maximum number of coprocessors the number of available master keys may be restricted.

The following table illustrates a configuration which would generate an inconsistency message by the check. In this scenario all 5 coprocessors are active. The AES, ECC and P11 master keys are available for use. The DES and RSA master keys are unavailable since they are not set on the relevant coprocessors. The DES master key is set on coprocessor G01 but not G00 and G02. The RSA master key is set on coprocessor G00 and G01 but not G02.

Table 64. CoProcessor/Master Key scenario

Cop \ MK	AES	DES	ECC	RSA	P11
G00	A	U	A	C	
G01	A	C	A	C	
G02	A	U	A	U	
SP04					A
SP05					A

The ICSF_MASTER_KEY_CONSISTENCY health check detects the inconsistency in master key states and generates a health check exception messages indicating that the states of the DES and RSA master keys are not consistent across the coprocessors. If the DES and RSA master keys were set on all three coprocessors then both master keys would be available for use.

When the health check is run, one of the following messages is generated:

The CSFH0014I message is generated if there are no problems.

The CSFH0015E message is generated if there is a master key inconsistency.

The CSFH0016E unable to process request. For example, an inconsistency in the AES master key would generate the following exception:[3]

ICSF_OPTIONS_CHECKS

Type: Status

Initial State: Active

Interval: One time

z/OS and ICSF releases the check applies to: ICSF FMID HCR7780, HCR7790, HCR77A0, HCR77A1, HCR77B0, and HCR77B1 running on z/OS V1R9, z/OS V1R10, z/OS V1R11, z/OS V1R12, z/OS V1R13, z/OS V2R1, or z/OS V2R2 with PTFs for APAR OA48452 applied, or ICSF FMID HCR77C0 and later running on z/OS V2R1 and later.

This check examines the value of some of the ICSF installation options that can affect the performance of your ICSF applications. The check reports whether the current setting matches the option setting supplied in the parameter or the default setting. Any option setting not specified in the parameter will be checked for the default described for that option.

Parameters accepted: The following parameters are supported to identify installation options to be checked:

CHECKAUTH(YES | NO)
Specifies that the check compare the current value of the CHECKAUTH option specified in the ICSF installations options data set. The CHECKAUTH option controls the SAF checking of authorized callers.

Default: NO

3.
```
CHECK(IBMICSF, ICSF_MASTER_KEY_CONSISTENCY)
START TIME: 09/23/2012 14:32:34.584930
CHECK DATE: 20120101 CHECK SEVERITY: MEDIUM

* Medium Severity Exception *

CSFH0015E The state of the AES master key is not consistent across all
coprocessors.

Explanation: The current value for the specified master key is not consistent
across the coprocessors. At least one coprocessor has the specified master
key in a state that is not in agreement with the other coprocessors.

System action: Alert the ICSF Administrator to determine the impact of the
current coprocessor states.

User response: Report this exception to the ICSF Administrator.

Administrator response: Refer to the ICSF Coprocessor Management and hardware
status panels. The state of the specified master key should match for all
Active or Online coprocessors. If problem is not resolved, contact the IBM
Support Center.
```

CKTAUTH(YES | NO)

Specifies that the check compare the current value of the CKTAUTH option specified in the ICSF installations options data set. The CKTAUTH option controls when CKDS record authentication codes are checked when a CKDS is loaded into the in-storage copy.

Note: The CKTAUTH option was deprecated in ICSF FMID HCR77A1 and specifying the option on an ICSF FMID HCR77A1 or newer release will have no effect on this check.

Default: NO

KEYAUTH(YES | NO)

Specifies that the check compare the current value of the KEYAUTH option specified in the ICSF installations options data set. The KEYAUTH option controls whether in-storage CKDS records have their record authentication code checked when the record is read by an application.

Note: The KEYAUTH option was deprecated in ICSF FMID HCR77A1 and specifying the option on an ICSF FMID HCR77A1 or newer release will have no effect on this check.

Default: NO

The following is an example of PARM specification:

```
PARM('CHECKAUTH(NO),CKTAUTH(NO),KEYAUTH(NO)')
```

The check is activated during the initialization of ICSF. The check is performed once at ICSF initialization.

When the ICSF_OPTIONS_CHECKS health check is run, the following messages are generated:

- Message CSFH0036I is an informational message that indicates all options match the specified values.
- Message CSFH0037E indicates that there are options that do not match the specified values.

ICSF_UNSUPPORTED_CCA_KEYS

Type: Status

Initial State: Active

Interval: One time

z/OS and ICSF releases the check applies to: ICSF FMID HCR77C0 and later running on z/OS V2R1 and later.

The Cryptographic Coprocessor Feature (CCF) supported cryptographic algorithms that are now not supported on newer Crypto Express features. Keys for these unsupported algorithms and services may have been stored in the CKDS and PKDS. The ICSF_UNSUPPORTED_CCA_KEYS health check lists the label of records in the active CKDS and PKDS with keys that are not supported by the current cryptographic features. The ICSF administrator can use the list to determine if the records should be deleted. The administrator can archive the records if the format of the CKDS and PKDS is the common record format (KDSR).

For the PKDS, all records with DSS public or private key tokens will be listed.

For the CKDS, all records with DES ANSI X9.17 keys and DES data-translation keys (key type DATAXLAT) will be listed.

The ICSF_UNSUPPORTED_CCA_KEYS check lists the labels of all records regardless of the state of the record. This means that records that are archived or inactive will appear in the list.

The check is activated during the initialization of ICSF. The check is performed every time ICSF is started.

When the ICSF_UNSUPPORTED_CCA_KEYS health check is run, the following messages are generated:

- Message CSFH0038I is an informational message that displays the health check header.
- Message CSFH0039I indicates that there are no records with unsupported keys.
- Message CSFH0040E indicates that there are records with unsupported keys.

For example:

```
CHECK(IBMICSF,ICSF_UNSUPPORTED_CCA_KEYS)
SYSPLEX:    LOCAL     SYSTEM: SY1
START TIME: 02/08/2016 09:13:24.949858
CHECK DATE: 20151001  CHECK SEVERITY: LOW

CSFH0038I Check for unsupported CCA cryptographic keys in CKDS and PKDS

Active CKDS: CSF.CKDS
---------------------------------------------------------
CSF.TEST.KEY.ABCD0001                          DATAXLAT
CSF.TEST.KEY.ABCD0004                          AKEK

Active PKDS: CSF.PKDS
---------------------------------------------------------
CSF.TEST.KEY.DSS.PVT.0001
CSF.TEST.KEY.DSS.PUB.0002
CSF.TEST.KEY.DSS.PUB.0005
CSF.TEST.KEY.DSS.PUB.0001

* Low Severity Exception *

CSFH0040E Unsupported CCA cryptographic keys in CKDS or PKDS were
found.

  Explanation:  This check detected keys in the active key data sets
    that are not supported with the cryptographic features in use. The
    label of the records are listed.

  System Action:  There is no effect on the system.

  Operator Response:  Contact the ICSF administrator.

  System Programmer Response:  Contact the ICSF administrator.

  Problem Determination:  If you are using the common record format
    (KDSR) of the CKDS and PKDS, you may be able to determine if the
    label is being used in an application.

    If you have key reference tracking enabled, you can tell if the
    label has been referenced by an application using the CSFMDR
    callable service to read the last reference data metadata of the
    record.
    If you don't have key reference tracking enabled, you can either
    enable key reference tracking or archive the record to see if the
    label is being used by an application.

  Source:  n/a

  Reference Documentation:
    z/OS Cryptographic Services ICSF Administrators Guide
    z/OS Cryptographic Services ICSF Application Programmers Guide

  Automation:  n/a

  Check Reason:  Detects CCA keys that are unsupported.

END TIME: 02/08/2016 09:13:24.954074  STATUS: EXCEPTION-LOW
```

ICSFMIG_DEPRECATED_SERV_WARNINGS

Type: Migration

Initial State: Inactive

Interval: Daily

z/OS and ICSF releases the check applies to: ICSF FMID HCR7790, HCR77A0, and HCR7A1 running on z/OS V1R11, z/OS V1R12, z/OS V1R13, and z/OS V2R1.

This is a migration check. If you are migrating to ICSF FMID HCR77A1 or a later release, you should run this check on your system before installing the new release of ICSF. The check detects the use of services which will not be supported in subsequent releases of ICSF. The check is not active when ICSF is started and must be activated to perform the check. Once activated the check will be performed on a daily basis.

The check output is obtained by selecting (s) the check on the Health Checker menu. If the check determines that deprecated services are being used then an exception is generated.

The check output is obtained by selecting (s) on the Health Checker menu:

```
CHECK(IBMICSF,ICSFMIG_DEPRECATED_SERV_WARNINGS)
START TIME: 05/20/2011 08:33:39.248906
CHECK DATE: 20110320  CHECK SEVERITY: LOW

* Low Severity Exception *

CSFH0011I Cryptographic Service CSFAKEX is currently used,
but support for this service is being removed in subsequent releases.

  Explanation:  The specified callable service is not being supported in
    subsequent releases, thus in the future workloads using the service
    may fail.

  System Action:  There is no effect on this system.

  Operator Response:  Report this exception to the System Programmer.

  System Programmer Response:  Alert the installation security
    Administrator and application/middleware administrators for this
    system.

  Problem Determination:  Investigate applications using this service
```

```
    and determine appropriate actions to remove or replace the use of
    this service.

  Source:  Integrated Cryptographic Service Facility (ICSF)

  Reference Documentation:  z/OS Cryptographic Services Integrated
    Cryptographic Service Facility: Application Programmers Guide
    (ICSF FMID HCR7790 and later).

  Automation:  n/a

  Check Reason:  Detects use of deprecated callable service.

END TIME: 05/20/2011 08:35:40.308973  STATUS: EXCEPTION-LOW
```

The deprecated services checked in this release are listed below. These are not supported on post zSeries 900 hardware.

- CSFAEGN
- CSFAKEX
- CSFAKIM
- CSFAKTR
- CSFATKN
- CSFCTT
- CSFCTT1
- CSFTCK
- CSFUDK
- CSFPKSC

ICSFMIG_MASTER_KEY_CONSISTENCY

Type: Migration

Initial State: Inactive

Interval: One Time

z/OS and ICSF releases the check applies to: ICSF FMID HCR7751, HCR7770, HCR7780, and HCR7790 running on z/OS V1R6, z/OS V1R7, z/OS V1R8, z/OS V1R9, z/OS V1R10, z/OS V1R11, z/OS V1R12, and z/OS V1R13 with PTFs for APAR OA39489 applied.

This is a migration check introduced in APAR OA39489. The check detects inconsistencies in the states of the cryptographic coprocessor master keys. The check is intended to warn the user of potential problems when migrating from pre-FMID HCR7780 releases of ICSF to the FMIDs HCR7780, HCR7790, or HCR77A0 releases of ICSF. The check is inactive when ICSF is started. When activated, it performs a one time check on the states of the coprocessor master keys. If a master key is not consistent across the available coprocessors, a problem condition is assumed and a health checker exception message is generated for the administrator's attention.

The following master key states are defined for use in describing this migration health check: available ('A'), correct ('C'), error ('E'), uninitialized ('U'), or not supported (-).

Available
Indicates that the master key matches the key used in the CKDS/PKDS and is available for use.

Correct
Indicates that the key matches the key used in the CKDS/PKDS, but is not available for use.

Error Indicates that the key does not match the key used in the CKDS/PKDS.

Uninitialized
Indicates that the key has not been set.

Table 65 and Table 66 illustrate a problem scenario. The pre-FMID HCR7780 releases of ICSF require a DES master key. For these releases, the G01 coprocessor is active since it has the DES master key set, but the G00 and G02 coprocessors are not active because they do not have the DES master key set. Because all four master keys are valid for the G01 coprocessor, all four master keys are available.

Table 65. Coprocessor/Master Key configuration on a pre-FMID HCR7780 system

Coprocessor \ Master Key	Coprocessor State	AES	DES	ECC	RSA
G00	Online	C	U	C	C
G01	Active	A	A	A	A
G02	Online	C	U	C	U

When a non-CCF system is migrated to the FMIDs HCR7780, HCR7790, or HCR77A0 releases of ICSF, the master states change. The migrated system will have all three coprocessors active; however, all master keys will not be available. The DES and RSA master keys will not be available. These keys are unavailable because they are not set on all active coprocessors.

Table 66. Coprocessor/Master Key configuration on a FMID HCR7780, HCR7790, or HCR77A0 release of ICSF

Coprocessor \ Master Key	Coprocessor State	AES	DES	ECC	RSA
G00	Active	A	U	A	C
G01	Active	A	C	A	C
G02	Active	A	U	A	U

The ICSFMIG_MASTER_KEY_CONSISTENCY health check detects problem states and generates health check exception messages indicating a problem with the DES and RSA master keys because these keys are not consistent across the coprocessors.

When the health check is run, one of the following messages is generated:

- The CSFH0014I message is generated if there are no problems.
- The CSFH0015E message is generated if there is a potential master key problem.
- The CSFH0016E message is generated if the system is unable to process the requested check.

ICSFMIG7731_ICSF_RETAINED_RSAKEY

Type: Migration

Initial State: Inactive

Interval: One Time

z/OS and ICSF releases the check applies to: ICSF FMID HCR7731 and later running on z/OS V1R9 and later.

This is a migration check. The check detects the presence of retained keys on the cryptographic coprocessors. Retained keys will not be supported in subsequent releases of ICSF. Existing retained keys will become unusable.

Retained keys are listed by coprocessor. The generated Health Checker report lists the coprocessor serial number and the retained key label. Existing retained keys must be replaced with RSA keys stored in the PKDS rather than retained on the coprocessor.

The check output is obtained by selecting (s) on the Health Checker menu:

```
CHECK(IBMICSF,ICSFMIG7731_ICSF_RETAINED_RSAKEY)
START TIME: 05/20/2011 08:16:29.689677
CHECK DATE: 20071201  CHECK SEVERITY: LOW
Coprocessor
  Serial     Retained key label
----------------------------------------------------------------------
93X06020  HCR7750.RKEY.RSA.CRT.1024MOD
93X06020  HCR7750.RKEY.RSA.CRT.1024MOD.SIGONLY

* Low Severity Exception *

CSFH0003E Cryptographic coprocessors were examined and found to
possess retained RSA Keys.

  Explanation:  Coprocessors online to this system were found to possess
    one or more retained RSA keys, implying retained RSA keys are
    potentially being used on this system. ICSF is deprecating its
    retained RSA key support.

  System Action:  There is no effect on the system.

  Operator Response:  Report this exception to the System Programmer.

  System Programmer Response:  Alert the installation security
    Administrator and application and middleware administrators for this
    system.

  Problem Determination:  Investigate the cryptographic services
    utilized by the workload executed on this system and determine which
    application and middleware products use retained RSA key services
    for key management use that would depend upon the key labels in the
    report. Develop an immediate strategy to remove any dependencies on
    creating new ICSF-supported retained RSA keys prior to migration to
    ICSF release level HCR7750, and an eventual strategy to remove any
    dependencies on ICSF-supported retained key interfaces.

  Source:  Integrated Cryptographic Service Facility (ICSF)

  Reference Documentation:  z/OS Cryptographic Services Integrated
    Cryptographic Service Facility: Systems Programmers Guide (HCR7750
    and later).
```

```
Automation:  n/a

Check Reason:  Detects use of retained RSA private keys.
```

ICSFMIG77A1_COPROCESSOR_ACTIVE

Type: Migration

Initial State: Inactive

Interval: One Time

z/OS and ICSF releases the check applies to: ICSF FMID HCR7770, HCR7780, HCR7790, and HCR77A0 running on z/OS V1R9, z/OS V1R10, z/OS V1R11, z/OS V1R12, z/OS V1R13, or z/OS V2R1 with PTFs for APAR OA42011 applied.

This is a migration check. If you are migrating to ICSF FMID HCR77A1 or a later release, you should run this check on your system before installing the new release of ICSF.

The migration check detects CCA cryptographic coprocessors with master keys that do not match the CKDS and PKDS. A coprocessor that has master keys that do not match the CKDS and PKDS will not become active when ICSF FMID HCR77A1 or later is started. This will affect the availability of coprocessors for cryptographic work.

Note: Coprocessors that have been deactivated will not be checked.

The method to decide which coprocessors become active has changed for ICSF FMID HCR77A1 and later releases. The master key verification pattern (MKVP) of the current master key register will be compared against the MKVPs in the header record of the CKDS and PKDS. If the MKVP is in the header record, the current master key must match that MKVP in order for the coprocessor to become active. This applies to all master keys that the coprocessor supports. When there is a MKVP in a key store and the coprocessor does not support that master key, it is ignored. When a MKVP is not in a key store, the master key is ignored. Note that if there are no MKVPs in any key store, the coprocessor will be active. Note that an initialized CKDS that has no MKVPs in the header record cannot be used on a system that has online coprocessors.

The check output is obtained by selecting (s) on the Health Checker menu:

When the health check is run, the following messages are generated:

The CSFH0017I message is generated if there are no CCA coprocessors.

The CSFH0018I message indicates the active key stores used in the check.

The CSFH0019I message is generated if there are no problems.

The CSFH0020E message is generated for each of the coprocessors that will not become active.

The CSFH0021E message is generated if the request could not be processed.

For example, the coprocessor installed at index 01 does not have the correct AES master key, the health check will generate the following exception:

```
CHECK(IBMICSF,ICSFMIG77A1_COPROCESSOR_ACTIVE)
START TIME: 09/23/2013 14:32:34.584930
CHECK DATE: 20120101 CHECK SEVERITY: MEDIUM
```

```
* Medium Severity Exception *

CSFH0018I: Active key stores: CKDS CSF.CKDS and PKDS CSF.PKDS.

CSFH0019E Coprocessor 01 serial number ssssssss has mismatched AES master keys.

Explanation: The coprocessor installed with index  nn will not become active when
ICSF FMID HCR77A1 or later is installed. The current type master key(s) loaded on
the coprocessor do not have the same value (as indicated by the master key
verification pattern (MKVP)) as stored in the CKDS or PKDS.

System action:  There is no effect on the system.

Operator response:  Contact the ICSF administrator.

ICSF Administrator response: The administrator should load the correct master
keys as indicated in the message using the ICSF master key entry panels or TKE.
The master keys are set using the SETMK panel utility on the Master Key Management
panel. Rerun this migration check after all master keys have been processed.
```

ICSFMIG77A1_TKDS_OBJECT

Type: Migration

Initial State: Inactive

Interval: One Time

z/OS and ICSF releases the check applies to: ICSF FMID HCR7770, HCR7780, HCR7790, and HCR77A0 running on z/OS V1R9, z/OS V1R10, z/OS V1R11, z/OS V1R12, z/OS V1R13, or z/OS V2R1 with PTFs for APAR OA42011 applied.

This is a migration check. If you are migrating to ICSF FMID HCR77A1 or a later release, you should run this check on your system before installing the new release of ICSF.

Note: If you do not have a Token Data Set (TKDS) with PKDS #11 objects in it, there is no need to run this check.

In the ICSF FMID HCR77A1 release, ICSF is introducing a common key data set record format for CCA key tokens and PKCS #11 tokens and objects. This new format of the record adds new fields for key utilization and metadata. Because of the size of the new fields, some exisitng PKCS #11 objects in the TKDS may cause ICSF to fail to start.

The problem exists for TKDS object records with large objects. The 'User data' field in the existing record cannot be stored in the new record format if the object size is greater that 32,520 bytes. The TKDSREC_LEN field in the record has the size of the object. If the 'User data' field is not empty and the size of the object is greater than 32,520 bytes, the TKDS cannot be loaded.

This migration check will detect any TKDS object that is too large to allow the TKDS to be loaded when ICSF is started.

The problem can be corrected by:
- Modifying the attributes of the object to make it smaller, if possible.
- Removing the information in the 'User data' field of the object. The 'User data' field must be all zeros for it to be ignored.

- Copying the object using PKCS #11 services and deleting the old object.
- Deleting the object.

Note: ICSF does not provide any interface to modify the 'User data' field in the TKDS object record. The field can only be modified by editing the record.

The TKDS object record is documented in the ICSF System Programmer's Guide.

When the health check is run, the following messages are generated:

The CSFH0023I message indicates the active TKDS that was checked.

The CSFH0024I message is generated if there are no TKDS objects that failed the check.

The CSFH0025E message is generated if there are TKDS objects that failed the check.

For example:

```
CHECK(IBMICSF,ICSFMIG77A1_TKDS_OBJECT)
START TIME: 04/18/2013 08:54:38.293403
CHECK DATE: 20130301  CHECK SEVERITY: MEDIUM

CSFH0023I Active Token Data Set: CSF.TKDS

The following TKDS objects will lose information:
SAMPLE.TOKEN                    00000006T
SAMPLE.TOKEN                    00000005T

* Medium Severity Exception *

CSFH0025E TKDS objects were found that have too much data.

Explanation: This message indicates which objects failed this check.
    The handle of each object is listed.

System action:  There is no effect on the system.

Operator response:  Contact the ICSF administrator.

System Programmer Response:  Contact the ICSF administrator.

Problem Determination:  n/a

Source:  n/a

Reference Documentation:  z/OS Cryptographic Services Integrated
  Cryptographic Service Facility: Writing PKCS #11 Applications.

Automation:  n/a

Check Reason:  Detects objects in the TKDS that will prevent
ICSF from loading the TKDS during initialization.
```

ICSFMIG77A1_UNSUPPORTED_HW

Type: Migration

Initial State: Inactive

Interval: One Time

z/OS and ICSF releases the check applies to: ICSF FMID HCR7770, HCR7780, HCR7790, and HCR77A0 running on z/OS V1R9, z/OS V1R10, z/OS V1R11, z/OS V1R12, z/OS V1R13, or z/OS V2R1 with PTFs for APAR OA42011 applied.

This is a migration check. If you are migrating to ICSF FMID HCR77A1 or a later release, you should run this check on your system before installing the new release of ICSF.

The ICSF FMID HCR77A1 release does not support IBM Eserver zSeries 800 and 900 systems. This migration check will indicate if your system is supported or not by ICSF FMID HCR77A1 and later releases.

When the health check is run, the following messages are generated:

The CSFH0022I message is generated if your system is not supported.

For example, the system zSeries 800 and 900:

```
CHECK(IBMICSF,ICSFMIG77A1_UNSUPPORTED_HW)
START TIME: 04/18/2013 09:12:47.938778
CHECK DATE: 20130301  CHECK SEVERITY: MEDIUM
* Medium Severity Exception *

CSFH0022E (ICSF,ICSFMIG77A1_UNSUPPORTED_HW):
Current processor (z800 or z900) will not be supported on a migration
to ICSF FMID HCR77A1. ICSF FMID HCR77A1 is planned to require IBM zSeries z890,
z990, or newer processors.

Explanation: The processor this check was executed on will not be
    supported by ICSF FMID HCR77A1.  ICSF FMID HCR77A1 will not start on
    zSeries 900 and 800 processors.  All releases of ICSF prior to ICSF FMID
    HCR77A1 support the zSeries 900 and 800 processors.

System action:  There is no effect on the system.

Operator response:  Contact the ICSF administrator.

System Programmer Response:  Contact the ICSF administrator.

Problem Determination:  n/a

Source:  n/a

Reference Documentation:  z/OS Cryptographic Services Integrated
  Cryptographic Service Facility: Overview.

Automation:  n/a

Check Reason:  Detects systems that ICSF no longer supports.
```

Appendix A. ICSF Panels

This appendix contains examples of ICSF panels.

ICSF Primary Menu panel

```
HCR77C0 ------------- Integrated Cryptographic Service Facility ---------

System Name: SY1                                Crypto Domain: 3
Enter the number of the desired option.

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

      Licensed Materials - Property of IBM
      5650-ZOS (C) Copyright IBM Corp. 1989, 2016.
      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.
```

CSFACF00 — Administrative Control Functions panel

```
 CSFACF00 ------------- ICSF Administrative Control Functions
 COMMAND ===>
          Active CKDS: CSF.CKDS
          Active PKDS: CSF.PKDS
          Active TKDS: CSF.TKDS

To change the status of a control, enter the appropriate character
(E - ENABLE, D - DISABLE) and press ENTER.

         Function                                 STATUS
         --------                                 ------
 .  Dynamic CKDS Access                           ENABLED
 .  PKA Callable Services                         ENABLED
 .  Dynamic PKDS Access                           ENABLED
```

CSFCKD20 — CKDS Operations panel

```
CSFCKD20 ---------------- ICSF – CKDS Operations ----------------
 COMMAND ===>


 Enter the number of the desired option.

   1  Initialize an empty CKDS
   2  REFRESH   -  Activate an updated CKDS
   3  Update an existing CKDS
   4  Update an existing CKDS and activate master keys
   5  Refresh and activate master keys

Enter the name of the CKDS below.

   CKDS ===>
```

CSFCKD30 — PKDS Operations panel

```
CSFCKD30 ---------------- ICSF - PKDS Operations  ----------------
COMMAND ===>


Enter the number of the desired option.

  1  Initialize an empty PKDS and activate master keys
        KDSR format? (Y/N) ===>
  2  REFRESH   -  Activate a PKDS
  3  Update an existing PKDS
  4  Update an existing PKDS and activate master keys
  5  Refresh and activate master keys

Enter the name of the PKDS below.

  PKDS ===>

Press ENTER to execute your option.
Press END   to exit to the previous menu.
```

CSFCMK10 — Reencipher CKDS panel

```
CSFCMK10 ----------------- ICSF - Reencipher CKDS ------------------
COMMAND ===>


To reencipher all CKDS entries from encryption under the current master key
to encryption under the new master key enter the CKDS names below.



   Input CKDS ===>

   Output CKDS ===>
```

CSFCMK12 — Reencipher PKDS panel

```
 CSFCMK12 ---------------- ICSF - Reencipher PKDS -------------
 COMMAND ===>


  To reencipher all PKDS entries from encryption under the old RSA master
  key and/or current ECC master keys to encryption under the current RSA
  master key and/or new ECC master key, enter the PKDS names below.


    Input PKDS   ===>

    Output PKDS ===>

Press ENTER to reencipher the PKDS.
Press END   to exit to the previous menu
```

CSFCMK20 — Change Master Key panel

```
 CSFCMK20 ------------- ICSF Change Master Key ------------
 COMMAND ===>


 Enter the name of the new CKDS below:

   New CKDS ===>

 When the master key is changed, the new CKDS will become active.
```

CSFCMK21 — Refresh PKA Cryptographic Key Data Set panel

```
 CSFCMK21 ---------- ICSF - Refresh PKA Cryptographic Key Data Set -------
 COMMAND ===>


 Enter the name of the new PKDS below.

    New PKDS ===>

Press ENTER to refresh the PKDS.
Press END   to exit to the previous menu
```

CSFCMK22 — Change Asymmetric Master Key panel

```
CSFCMK22 ---------- ICSF - Change Asymmetric Master Key -----------

Enter the name of the new PKDS below.

New PKDS ===>

When the master key is changed, the new PKDS will become active.
```

CSFCMK30 — Initialize a PKDS panel

```
CSFCMK30 ---------------- ICSF - Initialize a PKDS  ----------------
COMMAND ===>


Enter the name of the PKDS to be initialized below.

  PKDS ===>
```

CSFCMP00 — Coprocessor Management panel

```
CSFCMP00 ---------------- ICSF Coprocessor Management -------- Row 1 to 1 of 1

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

  CRYPTO    SERIAL
  FEATURE   NUMBER    STATUS               AES   DES   ECC   RSA   P11
  -------   --------  -------------------- ---   ---   ---   ---   ---
.   5C37    99EA6008  Master key incorrect   E     C     C     C
```

CSFMKM10 — Key Data Set Management panel

```
CSFMKM10 ---------------- ICSF – Key Data Set Management ----------------
OPTION ===>  1

Enter the number of the desired option.

  1  CKDS MK MANAGEMENT - Perform Cryptographic Key Data Set (CKDS)
                          functions including master key management
  2  PKDS MK MANAGEMENT - Perform Public Key Data Set (PKDS)
                          functions including master key management
  3  TKDS MK MANAGEMENT - Perform PKCS #11 Token Data Set (TKDS)
                          functions including master key management
  4  SET MK             - Set master key
```

CSFMKM20 — CKDS Management panel

```
CSFMKM20 ---------------- ICSF - CKDS Management  ----------------
OPTION ===>  1

Enter the number of the desired option.

 1  CKDS OPERATIONS   -  Initialize a CKDS, activate a different CKDS,
                         (Refresh), or update the header of a CKDS and make
                         it active
 2  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing a symmetric
                         master key
 3  CHANGE SYM MK     -  Change a symmetric master key and activate the
                         reenciphered CKDS
 4  COORDINATED CKDS REFRESH - Perform a coordinated CKDS refresh
 5  COORDINATED CKDS CHANGE MK - Perform a coordinated CKDS change master key
 6  COORDINATED CKDS CONVERSION - Convert the CKDS to use KDSR record format
 7  CKDS KEY CHECK    – Check keys in the active CKDS for format errors
```

CSFMKM30 — PKDS Management panel

```
CSFMKM30 ---------------- ICSF - PKDS Management  ----------------
OPTION ===>  1

Enter the number of the desired option.

 1  PKDS OPERATIONS   -  Initialize a PKDS, activate a different PKDS,
                         (Refresh), or update the header of a PKDS and make
                         it active
 2  REENCIPHER PKDS   -  Reencipher the PKDS
 3  CHANGE ASYM MK     - Change an asymmetric master key and activate the
                         reenciphered PKDS
 4  COORDINATED PKDS REFRESH - Perform a coordinated PKDS refresh
 5  COORDINATED PKDS CHANGE MK - Perform a coordinated PKDS change master key
 6  COORDINATED PKDS CONVERSION - Convert the PKDS to use KDSR record format
 7  PKDS KEY CHECK    – Check keys in the active PKDS for format errors

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

OPTION ===>
```

CSFMKV00 — Checksum and Verification Pattern panel

```
CSFMKV00 ------------ ICSF - Checksum and Verification and Hash Pattern -----
COMMAND ===>


Enter data below:

 Key Type      ===>                   (Selection panel displayed if blank)

 Key Value     ===> 51ED9CFA90716CFB  Input key value 1
               ===> 58403BFA02BD13E8  Input key value 2
               ===> 9B28AEFA8C47760F  Input key value 3 (AES & ECC & RSA Keys)
               ===> 0000000000000000  Input key value 4 (AES & ECC Keys only)

 Checksum         : 00                Check digit for key value
 Key Part VP      : 0000000000000000  Verification Pattern
 Key Part HP      : 0000000000000000  Hash Pattern
                  : 0000000000000000
```

CSFMKV10 — Key Type Selection panel

```
CSFMKV00 ------------ ICSF - Key Type Selection Panel ---- ROW 1 to 12 OF 12
COMMAND ===>

 Select one key type only
 KEY TYPE       DESCRIPTION
 AES-MK        AES Master Key
 DES-MK        DES Master key (16-byte)
   DES24-MK DES Master key (24-byte)
 ECC-MK        ECC Master key
 EXPORTER       Export key encrypting key
 IMP-PKA        Limited authority importer key
 IMPORTER       Import key encrypting key
 IPINENC       Input PIN encrypting key
 OPINENC       Output PIN encrypting key
 PINGEN        PIN generation key
 PINVER        PIN verification key
 RSA-MK        RSA Master key
**************************** BOTTOM OF DATA *****************************
```

CSFPMC20 — Pass Phrase MK/CKDS/PKDS Initialization

```
CSFPMC20 --------- ICSF - Pass Phrase MK/CKDS/PKDS Initialization ----------

ARE YOU SURE YOU WISH TO PROCEED WITH PASS PHRASE INITIALIZATION?

There are currently coprocessors with valid valid_master_key_types master
key(s). If you proceed with pass phrase initialization, the master key value(s)
May change.

If you wish to initialize new coprocessors only, return to the previous panel
and select the Add coprocessors action.

To proceed with pass phrase initialization, PKA callable services must be
disabled. Use the Administrative Control Functions utility to disable PKA
callable services.


Press ENTER to proceed with pass phrase initialization.
Press END to exit to the previous menu.
```

CSFPPM00 — Master Key Values from Pass Phrase panel

```
 CSFPPM00 ------------ ICSF - Master Key Values from Pass Phrase -----

Pass Phrase ( 16 to 64 characters)
 ==>_________________________________________________________________

Signature/Asymmetric-keys master key  : 0000000000000000
                                      : 0000000000000000
                                      : 0000000000000000

Key Management master key             : 0000000000000000
                                      : 0000000000000000
                                      : 0000000000000000
```

CSFRNG00 — ICSF Random Number Generator panel

```
 CSFRNG00 ---------------- ICSF - Random Number Generator -------------
 COMMAND ===>


 Enter data below:

   Parity Option  ===> RANDOM            ODD, EVEN, RANDOM
   Random Number1    : 0000000000000000  Random Number 1
   Random Number2    : 0000000000000000  Random Number 2
   Random Number3    : 0000000000000000  Random Number 3
   Random Number4    : 0000000000000000  Random Number 4
```

CSFSOP00 — Installation Options panel

```
 CSFSOP00 ----------- ICSF - Installation Options --------
 OPTION ===> 2

 Enter the number of the desired option above.

   1  OPTIONS  -  Display Installation Options
   2  EXITS    -  Display Installation exits and exit options
   3  SERVICES -  Display Installation Defined Services
```

CSFSOP30 — Installation Exits Display panel

```
CSFSOP30 ------ ICSF - Installation Exits Display ---- ROW 1 TO 18 OF 70
COMMAND ===>

ICSF NAME      LOAD MODULE      OPTIONS
-------------  -----------      -------
 CSFCKDS                        *** No Exit Name was specified ***
 CSFCKI                         *** No Exit Name was specified ***
 CSFCKM                         *** No Exit Name was specified ***
 CSFCONVX                       *** No Exit Name was specified ***
 CSFCPA                         *** No Exit Name was specified ***
 CSFCPE                         *** No Exit Name was specified ***
 CSFCSG                         *** No Exit Name was specified ***
 CSFCSV                         *** No Exit Name was specified ***
 CSFCTT2                        *** No Exit Name was specified ***
 CSFCTT3                        *** No Exit Name was specified ***
 CSFCVE                         *** No Exit Name was specified ***
 CSFCVT                         *** No Exit Name was specified ***
 CSFDCO                         *** No Exit Name was specified ***
 CSFDEC                         *** No Exit Name was specified ***
 CSFDEC1                        *** No Exit Name was specified ***
 CSFDKG                         *** No Exit Name was specified ***
 CSFDKM                         *** No Exit Name was specified ***
 CSFDKX                         *** No Exit Name was specified ***
 CSFDSG                         *** No Exit Name was specified ***
 CSFDSV                         *** No Exit Name was specified ***
 CSFDVPI                        *** No Exit Name was specified ***
 CSFECO                         *** No Exit Name was specified ***
 CSFEDC        USEREDC          NONE - Take no action, if this exit fails
```

CSFUTL00 — ICSF Utilities panel

```
CSFUTL00 ---------------- ICSF - Utilities --------------------------
OPTION ===>


Enter the number of the desired option.

  1  ENCODE        -  Encode data
  2  DECODE        -  Decode data
  3  RANDOM        -  Generate a random number
  4  CHECKSUM      -  Generate a checksum and verification and
                      hash pattern
  5  PPKEYS        -  Generate master key values from a pass phrase
  6  PKDSKEYS      -  Manage keys in the PKDS
  7  PKCS11 TOKEN  -  Management of PKCS11 tokens






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

OPTION ===>
```

Appendix B. Control Vector Table

Note: The Control Vectors used in ICSF are exactly the same as in CCA and the TSS publication.

The master key enciphers all keys operational on your system. A transport key enciphers keys that are distributed off your system. Prior to a master key or transport key enciphering a key, ICSF exclusive ORs both halves of the master key or transport key with a control vector. The same control vector is exclusive ORed to the left and right half of a master key or transport key.

Also, if you are entering a key part, ICSF exclusive ORs each half of the key part with a control vector prior to placing the key part into the CKDS.

Each type of key on ICSF (except the master key) has either one or two unique control vectors associated with it. The control vector that ICSF exclusive ORs the master key or transport key with depends on the type of key the master key or transport key is enciphering. For double-length keys, a unique control vector exists for each half of a specific key type. For example, there is a control vector for the left half of an input PIN-encrypting key, and a control vector for the right half of an input PIN-encrypting key.

If you are entering a key part into the CKDS, ICSF exclusive ORs the key part with the unique control vector or vectors associated with the key type. ICSF also enciphers the key part with two master key variants for a key part. One master key variant enciphers the left half of the key part, and another master key variant enciphers the right half of the key part. ICSF creates the master key variants for a key part by exclusive ORing the master key with the control vectors for key parts. These procedures protect key separation.

Table 67 displays the default value of the control vector that is associated with each type of key. For keys that are double-length, ICSF enciphers a unique control vector on each half.

Table 67. Default Control Vector Values

Key Type	Control Vector Value (Hex) Value for Single-length Key or Left Half of Double-length Key	Control Vector Value (Hex) Value for Right Half of Double-length Key
CIPHER	00 03 71 00 03 00 00 00	
CIPHER (double length)	00 03 71 00 03 41 00 00	00 03 71 00 03 21 00 00
CIPHERXI	00 0C 50 00 03 C0 00 00	00 0C 50 00 03 A0 00 00
CIPHERXO	00 0C 60 00 03 C0 00 00	00 0C 60 00 03 A0 00 00
CIPHERXL	00 0C 71 00 03 C0 00 00	00 0C 71 00 03 A0 00 00
CVARDEC	00 3F 42 00 03 00 00 00	
CVARENC	00 3F 48 00 03 00 00 00	
CVARPINE	00 3F 41 00 03 00 00 00	
CVARXCVL	00 3F 44 00 03 00 00 00	
CVARXCVR	00 3F 47 00 03 00 00 00	

Table 67. Default Control Vector Values (continued)

Key Type	Control Vector Value (Hex) Value for Single-length Key or Left Half of Double-length Key	Control Vector Value (Hex) Value for Right Half of Double-length Key
DATA	00 00 00 00 00 00 00 00	
DATA (internal)	00 00 7D 00 03 41 00 00	00 00 7D 00 03 21 00 00
DATA (external)	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
DATAC	00 00 71 00 03 41 00 00	00 00 71 00 03 21 00 00
DATAM generation key (external)	00 00 4D 00 03 41 00 00	00 00 4D 00 03 21 00 00
DATAM key (internal)	00 05 4D 00 03 00 00 00	00 05 4D 00 03 00 00 00
DATAMV MAC verification key (external)	00 00 44 00 03 41 00 00	00 00 44 00 03 21 00 00
DATAMV MAC verification key (internal)	00 05 44 00 03 00 00 00	00 05 44 00 03 00 00 00
DECIPHER	00 03 50 00 03 00 00 00	
DECIPHER (double-length)	00 03 50 00 03 41 00 00	00 03 50 00 03 21 00 00
DKYGENKY	00 71 44 00 03 41 00 00	00 71 44 00 03 21 00 00
ENCIPHER	00 03 60 00 03 00 00 00	
ENCIPHER (double-length)	00 03 60 00 03 41 00 00	00 03 60 00 03 21 00 00
EXPORTER	00 41 7D 00 03 41 00 00	00 41 7D 00 03 21 00 00
IKEYXLAT	00 42 42 00 03 41 00 00	00 42 42 00 03 21 00 00
IMP-PKA	00 42 05 00 03 41 00 00	00 42 05 00 03 21 00 00
IMPORTER	00 42 7D 00 03 41 00 00	00 42 7D 00 03 21 00 00
IPINENC	00 21 5F 00 03 41 00 00	00 21 5F 00 03 21 00 00
MAC	00 05 4D 00 03 00 00 00	
MAC (double-length)	00 05 4D 00 03 41 00 00	00 05 4D 00 03 21 00 00
MACVER	00 05 44 00 03 00 00 00	
MACVER (double-length)	00 05 44 00 03 41 00 00	00 05 44 00 03 21 00 00
OKEYXLAT	00 41 42 00 03 41 00 00	00 41 42 00 03 21 00 00
OPINENC	00 24 77 00 03 41 00 00	00 24 77 00 03 21 00 00
PINGEN	00 22 7E 00 03 41 00 00	00 22 7E 00 03 21 00 00
PINVER	00 22 42 00 03 41 00 00	00 22 42 00 03 21 00 00

Note:

1. The external control vectors for DATAC, double-length MAC generation and MAC verification keys are also referred to as data compatibility control vectors.

Appendix C. Supporting Algorithms and Calculations

This appendix shows various algorithms and calculations that are used in cryptographic systems.

Checksum Algorithm

To enter a key or a master key manually, you enter key parts. When you enter a key part, you enter two key part halves and a checksum for the key part. The checksum is a two-digit number you calculate using the key part and the checksum algorithm.

When you enter the key part and the checksum, ICSF calculates the checksum for the key part you entered. If the checksum you enter and the checksum ICSF calculates do not match, you did not enter the key part correctly and should reenter it. When you enter a key part, you need to calculate the checksum. You can use the checksum algorithm that is described in this appendix.

In the checksum algorithm, you use these operations:

- Sum Operation

  The addition table in Figure 145 on page 398 defines the sum operation. The sum of two hexadecimal digits i and j is the entry at the intersection of the column i and the row j. For example, the sum of A and 6 is C.

Sum	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	0	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	3	0	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	2	1	0	7	6	5	4	B	A	9	8	F	E	D	C
4	4	5	6	7	0	1	2	3	C	D	E	F	8	9	A	B
5	5	4	7	6	1	0	3	2	D	C	F	E	9	8	B	A
6	6	7	4	5	2	3	0	1	E	F	C	D	A	B	8	9
7	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8
8	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	9	8	B	A	D	C	F	E	1	0	3	2	5	4	7	6
A	A	B	8	9	E	F	C	D	2	3	0	1	6	7	4	5
B	B	A	9	8	F	E	D	C	3	2	1	0	7	6	5	4
C	C	D	E	F	8	9	A	B	4	5	6	7	0	1	2	3
D	D	C	F	E	9	8	B	A	5	4	7	6	1	0	3	2
E	E	F	C	D	A	B	8	9	6	7	4	5	2	3	0	1
F	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Figure 145. Addition Table

- Shift Operation

  The shift table in Figure 146 defines the shift operation. The shift of digit i is denoted by H(i). For example, the shift of 5 is H(5) = E.

i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
H(i)	0	C	1	D	2	E	3	F	4	8	5	9	6	A	7	B

Figure 146. Shift Table

In this description of the algorithm, the two hexadecimal digits of the checksum are represented by P1 and P2 for the set of 32 hexadecimal digits D(1,2,.....,32). The letter i represents the increment.

To calculate the checksum, use this algorithm:

1. Set i = 0, and set P1 and P2 = 0 (hexadecimal).
2. Let P1 = Sum of P1 and D(i + 1). Let P2 = Sum of P2 and D(i + 2).
3. Let P1 = H(P1). Let P2 = H(P2).
4. Let i = i + 2. If i < 32, go to step 2; otherwise, go to step 5.
5. P1 equals the first checksum digit. P2 equals the second checksum digit.

Algorithm for calculating a verification pattern

To enter a master key or operational key manually, you enter key parts. When you enter a key part, ICSF displays a verification pattern for that key part on a panel. To verify that you entered the key part correctly, you can use the value of the key part you enter to calculate the verification pattern. Check that the verification pattern you calculate matches the verification ICSF calculates.

To calculate this verification pattern for DES operational keys and the 16-byte DES master key, use this algorithm:

1. If the key part is an operational key part, exclusive OR the key part with the control vector for the key part's key type. See Appendix B, “Control Vector Table,” on page 395, for a listing of control vectors by key type. If the key part is a master key part, do not exclusive OR it with a control vector.
2. Use the DES algorithm to encrypt the left half of the key part (either master key part or modified operational key part) under the key `4545 4545 4545 4545`.
3. Exclusive OR the result of step 2 with the left half of the key part.
4. Use the result of step 3 as the DES key in the DES algorithm to encrypt the right half of the key part.
5. Exclusive OR the result of step 4 with the right half of the key part.

The resulting 64-bit value is the verification pattern.

To calculate this verification pattern for the 24-byte DES master key, use this algorithm:

1. Appending X'01' to the clear key value of 24-byte master key (01 || key value)
2. Generating the SHA-1 hash of the 25-byte string

The first eight bytes of the hash is the verification pattern.

The verification pattern for the master key appears on the Coprocessor Selection and Hardware Status panels. If a master key register is full, the panels display the master key verification pattern. The verification patterns for two identical master keys are the same. You can use the verification patterns to verify that master keys in two different key storage units are the same.

ICSF records a master key verification pattern in the SMF record when you enter a master key part or activate a master key. The ICSF SMF record also records a verification pattern when you enter an operational key part.

AES and ECC master key verification pattern algorithm

The AES and ECC master key verification pattern is calculated by:

1. Appending X'01' to the clear key value of 32-byte master key (01 || key value).
2. Generating the SHA-256 hash of the 33-byte string.

The first eight bytes of the hash is the verification pattern.

Pass Phrase Initialization master key calculations

The values for the master keys are calculated in this manner:

1. ICSF appends a two-byte constant, X'AB45', to the pass phrase, and generates the MD5 hash for the string by using an initial hash value of X'23A0BE487D9BD32003424FAAA34BCE00'. The first eight bytes of the result of this calculation become the last eight bytes of the RSA master key.

2. ICSF appends a four-byte constant, X'551B1B1B', to the pass phrase, and generating the MD5 hash for the string using the hash that results from Step 1 as the initial hash value. For system with a 16-byte DES master key, the output of this step is the master key. For a 24-byte DES master key, the last eight bytes of the results of step 1 is pre-appended to output of this step to get the master key value.
3. ICSF appends a three-byte constant, X'2A2A88', to the pass phrase and generates the MD5 hash for the string using the output hash of Step 2 as the initial hash value. The result of this calculation becomes the first 16 bytes of RSA master key.
4. ICSF appends a one-byte constant, X'94' to the pass phrase, and generates the MD5 hash for the string using the output hash of Step 3 as the initial hash value. The result of this calculation is not used, but is required for compatibility.
5. ICSF appends a five-byte constant X'C1C5E2D4D2' to the pass phrase, and generates the SHA-256 hash for the string using the output hash of Step 4 as the initial hash value. The result of this calculation becomes the 32-byte AES master key.
6. ICSF appends a seven-byte constant X'C5D3D3C9D7E2C5' to the pass phrase and generates the SHA-256 hash for the string using the output hash of Step 5 as the initial hash value. The result of this calculation becomes the 32-byte ECC master key.

The MDC–4 Algorithm for Generating Hash Patterns

The MDC-4 algorithm calculation is a one-way cryptographic function that is used to compute the hash pattern of a key part. MDC uses encryption only, and the default key is 5252 5252 5252 5252 2525 2525 2525 2525.

Notations Used in Calculations

The MDC calculations use this notation:

eK(X)	Denotes DES encryption of plaintext X using key K
||	Denotes the concatenation operation
XOR	Denotes the exclusive-OR operation
:=	Denotes the assignment operation
T8<1>	Denotes the first 8–byte block of text
T8<2>	Denotes the second 8–byte block of text, and so on
KD1, KD2, IN1, IN2, OUT1, OUT2	Denote 64–bit quantities

MDC-1 Calculation

The MDC-1 calculation, which is used in the MDC-4 calculation, consists of this procedure:

```
MDC-1 (KD1, KD2, IN1, IN2, OUT1, OUT2);
   Set KD1mod := set bit 1 and bit 2 of KD1 to "1" and "0", respectively.
   Set KD2mod := set bit 1 and bit 2 of KD2 to "0" and "1", respectively.
   Set F1 := IN1 XOR eKD1mod(IN1)
   Set F2 := IN2 XOR eKD2mod(IN2)
   Set OUT1 := (bits 0..31 of F1) || (bits 32..63 of F2)
   Set OUT2 := (bits 0..31 of F2) || (bits 32..63 of F1)
End procedure
```

MDC-4 Calculation

The MDC-4 calculation consists of this procedure:

```
MDC-4 (n, text, KEY1, KEY2, MDC);
   For i := 1, 2,...n do
      Call MDC-1(KEY1,KEY2,T8<i>,T8<i>,OUT1,OUT2)
      Set KEY1int := OUT1
      Set KEY2int := OUT2
      Call MDC-1(KEY1int,KEY2int,KEY2,KEY1,OUT1,OUT2)
      Set KEY1 := OUT1
      Set KEY2 := OUT2
   End do
   Set output MDC := (KEY1 || KEY2)
End procedure
```

Appendix D. PR/SM Considerations during Key Entry

If you use logical partition (LPAR) mode provided by the Processor Resource/System Manager (PR/SM), you may have additional considerations when performing these tasks:

- Entering keys
- Displaying hardware status
- Using the public key algorithm
- Using a TKE Workstation

These additional considerations depend on your processor hardware. For example, LPAR mode permits you to have multiple logical partitions and each logical partition (LP) can have access to the crypto CP for key entry. Therefore, at any given time, multiple LPs can perform key entry procedures.

This appendix gives some basic information on using ICSF in LPAR mode.

Allocating Cryptographic Resources to a Logical Partition

Logical Partitions (LPs) operate independently but can share access to the same cryptographic coprocessor, just as they can share access to I/O devices and any other central processor resources. When you activate the LP, you can specify which cryptographic functions are enabled for that LP. The cryptographic resources available to the LP and the way you allocate them to the LP depends on the server or processor your are using.

Allocating Resources

Dynamically enabling use of a new coprocessor or accelerator to a partition requires that:

- At least one usage domain index be defined to the logical partition.
- The usage domain list is a subset of the control domain list.
- The cryptographic coprocessor number or numbers be defined in the partition Candidate list.

The same usage domain index may be defined more than once across multiple logical partitions. However, the cryptographic coprocessor number coupled with the usage domain index specified must be unique across all active logical partitions.

The same cryptographic coprocessor number and usage domain index combination may be defined for more than one logical partition. In such a configuration, only one of the logical partitions can be active at any time. This may be used, for example, to define a configuration for backup situations.

Table 68 on page 404 illustrates a simplified configuration map.

Each row identifies a logical partition and each column a cryptographic coprocessor, installed or in plan. Each cell, indicates the Usage Domain Index number or numbers planned to be assigned to the partition in its image profile (it is recommended to work from a spreadsheet). There is a potential conflict when, for a given row, different cells contain more than once the same domain number.

Table 68. Planning LPARs domain and cryptographic coprocessor. Planning LPARs domain and cryptographic coprocessor

coprocessor ID	**AP0**	**AP1**	**AP2**	**AP3**	**AP4**	**AP5**	**AP6**	**...**
LPAR lp0	0	0				0	0	
LPAR lp1			0	0	0			
LPAR lp2	0	0	0	0	0			
LPAR lp4	4 14	4 14	4 14	4 14	4 14	4 14	4 14	
LPAR lp5				1	1	1	1	
.../...								

Within a row, the domain index number or numbers specified are identical because the domain index applies to all cryptographic coprocessors selected in the partition Candidate list. In the example:

- Logical partitions lp0 and lp1 use domain 0 but are assigned different cryptographic coprocessors. The combination domain number and cryptographic coprocessor number is unique across partitions. Both partitions lp0 and lp1 can both be active at the same time.
- Logical partition lp4 uses domain 4 and 14. Since no other partition uses the same domain numbers, there is no conflict.
- Logical partition lp5 uses domain 1 and no other partition uses the same domain number. Again, there is no conflict.
- Logical partitions lp2 use domain 0, on the set of cryptographic coprocessors already used by lp0 and lp1. Partition lp2 cannot be active concurrently with lp0 or lp1. However, this may be a valid configuration to cover for backup situations.

Entering the Master Key or Other Keys in LPAR Mode

To perform key entry from the TKE workstation, you must use a logical partition that already has key entry enabled.

In certain situations, ICSF clears the master key registers so the master key value is not disclosed. ICSF clears the master keys in all the logical partitions. The CKDSs and PKDSs are still enciphered under the master keys. To recover the keys in the CKDSs and PKDSs, you must reenter and activate the master keys used on your system.

To restore the master keys, first ensure that key entry is enabled for all usage domain indexes for which you need to reenter the master keys. Since multiple domains can have key entry enabled, the domains may already be enabled. Reenter and activate the master key for all usage domain indexes. You can do this either through the Clear Master Key Part Entry panels or the TKE workstation.

Reusing or Reassigning a Domain

In the course of business, you may find it necessary to reuse or reassign a domain that is currently active. If this is the case, there are several steps to perform. It is a good security practice to zeroize the domain secrets, which includes retained keys and master keys.

Run the retained key delete service in the domain to remove them.

You can zeroize the master key with the TKE workstation or with TSO panels. For information on the TKE process, see *z/OS Cryptographic Services ICSF TKE Workstation User's Guide*.

Appendix E. CCA access control points and ICSF utilities

For information about PKCS #11 access control points, see 'PKCS #11 Coprocessor Access Control Points' in *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

Access to utilities that are executed on the CCA coprocessor is through access control points (ACPs) in the ICSF role. To execute utilities on the coprocessor, access control points must be enabled for each service in the ICSF role. The TKE workstation allows you to enable or disable access control points.

A new or a zeroized coprocessor (or domain) comes with an initial set of access control points (ACPs) that are enabled by default. The table of access control points lists the default setting of each access control point.

When a firmware upgrade is applied to an existing cryptographic coprocessor, the upgrade may introduce new ACPs.

- If a TKE workstation has been used to manage a cryptographic coprocessor, the firmware upgrade does not retroactively enable the new ACPs. These ACPs must be enabled via the TKE (or subsequent zeroize) in order to utilize the new support they govern.
- If a TKE workstation has not been used to manage a cryptographic coprocessor, the firmware upgrade retroactively updates the new ACPs that would be enabled by default.

Note: Access control points for ICSF callable services are listed in appendix G of *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Access Control Points

If an access control point is disabled, the corresponding ICSF utility will fail during execution with an access denied error.

The table includes the following columns:

Name The descriptive name of the access control point. This is the name used when displaying the ICSF role from the ICSF Coprocessor Management panel.

Utility The utility that requires this access control point to be enabled for operation. The name is the CSFSERV profile name that controls the utility

Offset The hexadecimal offset, or access-control-point code, for the control in the domain role in the coprocessor.

Usage The following abbreviations and symbols are used in this table:

AE - Always enabled, cannot be disabled.
ED - Enabled by default.
DD - Disabled by default.
SC - Usage of this access control point requires special consideration.

Table 69. Access control points and associated utilities

Name	Utility	Offset (Hex)	Usage
Authorize UDX	CSFUDM	0240	AE
AES Master Key - Clear new master key register	CSFDKCS	0124	ED
AES Master Key - Combine key parts	CSFDKCS	0126	ED
AES Master Key - Load first key part	CSFDKCS	0125	ED
AES Master Key - Set master key	CSFDKCS	0128	AE
CKDS Conversion2 - Allow use of REFORMAT	CSFCNV2	014C	ED
CKDS Conversion2 - Allow wrapping override keywords	CSFCNV2	0146	AE
CKDS Conversion2 - Convert from enhanced to original	CSFCNV2	0147	ED
DES Master Key - Clear new master key register	CSFDKCS	0032	ED
DES Master Key - Combine key parts	CSFDKCS	0019	ED
DES Master Key - Load first key part	CSFDKCS	0018	ED
DES master key – 24-byte key	CSFDKCS	0330	DD, SC
DES Master Key - Set master key	CSFDKCS	001A	AE
ECC Master Key - Clear new master key register	CSFDKCS	031F	ED
ECC Master Key - Combine key parts	CSFDKCS	0321	ED
ECC Master Key - Load first key part	CSFDKCS	0320	ED
ECC Master Key - Set master key	CSFDKCS	0322	AE
Operational Key Load	CSFOPKL	0309	ED
Operational Key Load - Variable-Length Tokens	CSFOPKL	029E	ED
PCF CKDS Conversion - Allow wrapping override keywords	CSFCONV	0148	ED
PCF CKDS Conversion Program	CSFCONV	0303	ED
Reencipher CKDS	CSFRENC	001E	AE
Reencipher CKDS2	CSFRENC	00F0	AE
Reencipher PKDS	CSFPKDR	0241	AE
RSA Master Key - Clear new master key register	CSFDKCS	0060	ED
RSA Master Key - Combine key parts	CSFDKCS	0054	ED
RSA Master Key - Load first key part	CSFDKCS	0053	ED
RSA Master Key - Set master key	CSFDKCS	0057	AE

Note: When reenciphering the CKDS, the **CKDS Conversion2 - Allow wrapping override keywords** access control must be enabled.

Appendix F. Callable services affected by key store policy

This table provides application programmers guidance on parameters covered by the key store policy controls.

Only the names of the 31-bit versions of the callable services are listed. However, 64-bit versions of the callable services and the ALET qualified versions of the services are also covered by the key store policy. The callable services that are affected by the TOKEN_CHECK key store policy controls are in the table below.

Table 70. Callable services and parameters affected by key store policy

ICSF callable service	31-bit name	Parameter checked
Authentication Parameter Generate	CSNBAPG	inbound_PIN_encrypting_key_identifier AP_encrypting_key_identifier
Cipher Text Translate2	CSNBCTT2	key_identifier_in key_identifier_out
Clear PIN Encrypt	CSNBCPE	PIN_encrypting_key_identifier
Clear PIN Generate	CSNBPGN	PIN_generation_key_identifier
Clear PIN Generate Alternate	CSNBCPA	PIN_encryption_key_identifier PIN_generation_key_identifier
Control Vector Translate	CSNBCVT	KEK_key_identifier source_key_token array_key_left array_key_right
CVV Key Combine	CSNBCKC	key_a_identifier key_b_identifier
Cryptographic Variable Encipher	CSNBCVE	c_variable_encrypting_key_identifier
Data Key Export	CSNBDKX	source_key_identifier exporter_key_identifier
Data Key Import	CSNBDKM	source_key_token importer_key_identifier
Decipher	CSNBDEC	key_identifier
Derive ICC MK	CSNBDCM	issuer_master_key_identifier transport_key_identifier
Derive Session Key	CSNBDSK	master_key_identifier **Note:** ICC master keys derived from issuer master keys are affected by key store policy before they are used to derive session keys.
Digital Signature Generate	CSNDDSG	PKA_private_key_identifier
Digital Signature Verify	CSNDDSV	PKA_public_key_identifier
Diversified Key Generate	CSNBDKG	generating_key_identifier generated_key_identifier
Diversified Key Generate2	CSNBDKG2	generating_key_identifier

Table 70. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
DK Deterministic PIN Generate	CSNBDDPG	PIN_generation_key_identifier PRW_key_identifier PIN_print_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK Migrate PIN	CSNBDMP	IPINENC_key_identifier PRW_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PAN Modify in Transaction	CSNBDPMT	CMAC_FUS_key_identifier IPIN_encryption_key_identifier PRW_key_identifier new_PRW_key_identifier
DK PAN Translate	CSNBDPT	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PIN Change	CSNBDPC	PRW_MAC_key_identifier cur_IPIN_encryption_key_identifier new_IPIN_encryption_key_identifier script_key_identifier script_MAC_key_identifier new_PRW_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PIN Verify	CSNBDPV	PRW_MAC_key_identifier IPIN_encryption_key_identifier
DK PRW Card Number Update	CSNBDPNU	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PRW CMAC Generate	CSNBDPCG	CMAC_FUS_key_identifier
DK Random PIN Generate	CSNBDRPG	PRW_MAC_key_identifier PIN_print_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK Regenerate PRW	CSNBDRP	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
ECC Diffie-Hellman	CSNDEDH	private_key_identifier private_KEK_key_identifier public_key_identifier output_KEK_key_identifier

Table 70. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
EMV Scripting Service	CSNBESC	issuer_integrity_master_key_identifier issuer_confidentiality_master_key_identifier new_PIN_encrypting_key_identifier current_PIN_encrypting_key_identifier **Note:** ICC master keys derived from issuer master keys are affected by key store policy before they are used to derive session keys.
EMV Transaction Service	CSNBEAC	issuer_master_key_identifier issuer_ARPC_master_key_identifier **Note:** ICC master keys derived from issuer master keys are affected by key store policy before they are used to derive session keys.
EMV Verification Functions	CSNBEVF	key_identifier **Note:** ICC master keys derived from issuer master keys are affected by key store policy before they are used to derive session keys.
Encipher	CSNBENC	key_identifier
Encrypted PIN Generate	CSNBEPG	PIN_generating_key_identifier outbound_PIN_encrypting_key_identifier
Encrypted PIN Translate	CSNBPTR	input_PIN_encrypting_key_identifier output_PIN_encrypting_key_identifier
Encrypted PIN Translate Enhanced	CSNBPTRE	input_PIN_key_identifier output_PIN_key_identifier PAN_key_identifier
Encrypted PIN Verify	CSNBPVR	input_PIN_encrypting_key_identifier PIN_verifying_key_identifier
FPE Decipher	CSNBFPED	key_identifier
FPE Encipher	CSNBFPEE	key_identifier
FPE Translate	CSNBFPET	input_key_identifier output_key_identifier
Generate Issuer MK	CSNBGIM	transport_key_identifier
HMAC Generate	CSNBHMG	key_identifier
HMAC Verify	CSNBHMV	key_identifier
Key Encryption Translate	CSNBKET	KEK_key_identifier
Key Export	CSNBKEX	source_key_identifier exporter_key_identifier
Key Generate	CSNBKGN	KEK_key_identifier_1 KEK_key_identifier_2
Key Import	CSNBKIM	source_key_token importer_key_identifier
Key Test	CSNBKYT	key_identifier
Key Test2	CSNBKYT2	key_identifier
Key Test Extended	CSNBKYTX	key_identifier kek_key_identifier
Key Translate	CSNBKTR	input_KEK_key_identifier output_KEK_key_identifier

Table 70. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
Key Translate2	CSNBKTR2	input_key_token input_KEK_identifier output_KEK_identifier
MAC Generate	CSNBMGN	key_identifier
MAC Generate2	CSNBMGN2	key_identifier
MAC Generate2 (with ALET)	CSNBMGN3	key_identifier
MAC Verify	CSNBMGN	key_identifier
MAC Verify2	CSNBMVR2	key_identifier
MAC Verify2 (with ALET)	CSNBMVR3	key_identifier
Multiple Secure Key Import	CSNBSKM	key_encrypting_key_identifier
PIN Change/Unblock	CSNBPCU	authentication_issuer_master_key_identifier encryption_issuer_master_key_identifier new_reference_PIN_key_identifier current_reference_PIN_key_identifier
PKA Decrypt	CSNDPKD	PKA_key_identifier
PKA Encrypt	CSNDPKE	PKA_key_identifier
PKA Key Generate	CSNDPKG	transport_key_identifier
PKA Key Import	CSNDPKI	importer_key_identifier
PKA Key Translate	CSNDPKT	source_key_identifier source_transport_key_identifier target_transport_key_identifier
PKA Key Token Change	CSNDPKTC	key_identifier
PKA Public Key Extract	CSNDPKX	source_key_identifier target_public_key_token
Prohibit Export	CSNBPEX	key_identifier
Prohibit Export Extended	CSNBPEXX	source_key_token kek_key_identifier
Recover PIN From Offset	CSNBPFO	PIN_encryption_key_identifier PIN_generation_key_identifier
Remote Key Export	CSNDRKX	trusted_block_identifier transport_key_identifier importer_key_identifier source_key_identifier
Restrict Key Attribute	CSNBRKA	key_identifier
Secure Key Import	CSNBSKI	importer_key_identifier key_identifier
Secure Messaging for Keys	CSNBSKY	input_key_identifier key_encrypting_key_identifier secmsg_key_identifier
Secure Messaging for PINs	CSNBSPN	PIN_encrypting_key_identifier secmsg_key_identifier

Table 70. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
SET Block Compose	CSNDSBC	RSA_public_key_identifier DES_key_block RSA_OAEP_block
SET Block Decompose	CSNDSBD	RSA_private_key_identifier DES_key_block (one or two tokens)
Symmetric Algorithm Decipher	CSNBSAD	key_identifier
Symmetric Algorithm Encipher	CSNBSAE	key_identifier
Symmetric Key Export	CSNDSYX	DATA_key_identifier RSA_public_key_identifier
Symmetric Key Export with Data	CSNDSXD	source_key_identifier RSA_public_key_identifier
Symmetric Key Generate	CSFSYG	key_encrypting_key_identifier RSA_public_key_identifier DES_enciphered_key_token
Symmetric Key Import	CSNDSYI	RSA_enciphered_key RSA_private_key_identifier
Symmetric Key Import2	CSNDSYI2	RSA_private_key_identifier
Transaction Validation	CSNBTRV	transaction_key_identifier
Trusted Block Create	CSNDTBC	input_block_identifier transport_key_identifier
TR-31 Export	CSNBT31X	source_key_identifier unwrap_kek_identifier wrap_kek_identifier
TR-31 Import	CSNBT31I	unwrap_kek_identifier wrap_kek_identifier
Unique Key Derive	CSNBUKD	base_derivation_key_identifier transport_key_identifier
VISA CVV Service Generate	CSNBCSG	CVV_key_A_Identifier CVV_key_B_Identifier
VISA CVV Service Verify	CSNBCSV	CVV_key_A_Identifier CVV_key_B_Identifier

The callable services that are affected by the no duplicates key store policy controls are listed in the table below.

Table 71. Callable services that are affected by the no duplicates key store policy controls

ICSF callable service	31-bit name	Parameter checked
Key Part Import	CSNBKPI	key_identifier
Key Record Write	CSNBKRW	key_token
PKA Key Generate	CSNDPKG/CSNFPKG	generated_key_token
PKA Key Import	CSNDPKI/CSNFPKI	source_key_identifier
PKDS Key Record Read2	CSNDKRR2	token
PKDS Record Create	CSNDKRC/CSNFKRC	token

Table 71. Callable services that are affected by the no duplicates key store policy controls (continued)

ICSF callable service	31-bit name	Parameter checked
PKDS Record Read	CSNDKRR	token
PKDS Record Write	CSNDKRW	key_token
Trusted Block Create	CSNDTBC	input_block_identifier

Summary of Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions

For services that are passed a label, the key store policy will not affect the SAF check, so only Granular Keylabel Access Controls and CSNDSYX Access Controls will have an effect:

Table 72. Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions (label)

	No CSNDSYX Access Controls for algorithm	CSNDSYX Access Controls for algorithm	No Granular Keylabel Access Controls	Granular Keylabel Access Controls
CSNDSYX: DATA key identifier	Label SAF check is done against CSFKEYS	Label SAF check is done against XCSFKEY	n/a	n/a
CSNDSYX: RSA key identifier and all other services passed a label	n/a	n/a	Label SAF check is done against CSFKEYS for READ access	Label SAF check is done against CSFKEYS for appropriate access

For services that are passed a token:

Table 73. Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions (token)

	No KSP	KSP / No CSNDSYX Access Controls for algorithm	KSP / CSNDSYX Access Controls for algorithm	KSP / No Granular Keylabel Access Controls	KSP / Granular Keylabel Access Controls
CSNDSYX: DATA key identifier	No SAF check is done	KSP SAF checks are done against CSFKEYS	KSP SAF checks are done against XCSFKEY	n/a	n/a
CSNDSYX: RSA key identifier and all other services passed a label	No SAF check is done	n/a	n/a	KSP SAF checks are done against CSFKEYS	KSP SAF checks are done against CSFKEYS

Note: The levels used by Granular Keylabel Access Controls will also be applied to KSP checks (that is, if the CKDS labels matching a token were checked with UPDATE access, CSF-CKDS-DEFAULT will also be checked with UPDATE access)

Appendix G. Callable services that trigger reference date processing

This table provides guidance for application programmers on parameters that will trigger reference date processing. For all parameters, reference date processing will only be triggered when a label (CKDS or PKDS) or object handle (TKDS) is provided. See the description of each parameter in its respective callable service for details.

Only the names of the 31-bit versions of the callable services are listed. However, 64-bit versions of the callable services and the ALET qualified versions of the services are also covered by reference date processing.

Table 74. Callable services and parameters that trigger reference date processing

ICSF callable service	31-bit name	Parameter checked
Authentication Parameter Generate	CSNBAPG	inbound_PIN_encrypting_key_identifier AP_encrypting_key_identifier
CKDS Key Record Read	CSNBKRR	key_label
CKDS Key Record Read2	CSNBKRR2	key_label
CVV Key Combine	CSNBCKC	key_a_identifier key_b_identifier
Ciphertext Translate2	CSNBCTT2	key_identifier_in key_identifier_out
Clear PIN Encrypt	CSNBCPE	PIN_encrypting_key_identifier
Clear PIN Generate	CSNBPGN	PIN_generating_key_identifier
Clear PIN Generate Alternate	CSNBCPA	PIN_encryption_key_identifier PIN_generation_key_identifier
Control Vector Translate	CSNBCVT	KEK_key_identifier array_key_left array_key_right
Cryptographic Variable Encipher	CSNBCVE	c_variable_encrypting_key_identifier
Data Key Export	CSNBDKX	source_key_identifier exporter_key_identifier
Data Key Import	CSNBDKM	importer_key_identifier
Decipher	CSNBDEC	key_identifier
Derive ICC MK	CSNBDCM	issuer_master_key_identifier transport_key_identifier
Derive Session Key	CSNBDSK	master_key_identifier
Digital Signature Generate	CSNDDSG	private_key_identifier
Digital Signature Verify	CSNDDSV	PKA_public_key_identifier
Diversified Key Generate	CSNBDKG	generating_key_identifier key_identifier

Table 74. Callable services and parameters that trigger reference date processing (continued)

ICSF callable service	31-bit name	Parameter checked
Diversified Key Generate2	CSNBDKG2	generating_key_identifier
DK Deterministic PIN Generate	CSNBDDPG	PIN_generation_key_identifier PRW_key_identifier PIN_print_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK Migrate PIN	CSNBDMP	IPIN_encryption_key_identifier PRW_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PAN Modify in Transaction	CSNBDPMT	CMAC_FUS_key_identifier IPIN_encryption_key_identifier PRW_key_identifier new_PRW_key_identifier
DK PAN Translate	CSNBDPT	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PIN Change	CSNBDPC	PRW_key_identifier cur_IPIN_encryption_key_identifier new_IPIN_encryption_key_identifier script_key_identifier script_MAC_key_identifier new_PRW_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PIN Verify	CSNBDPV	PRW_key_identifier IPIN_encryption_key_identifier
DK PRW CMAC Generate	CSNBDPCG	CMAC_FUS_key_identifier
DK PRW Card Number Update	CSNBDPNU	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK Random PIN Generate	CSNBDRPG	PRW_key_identifier PIN_print_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK Regenerate PRW	CSNBDRP	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
ECC Diffie-Hellman	CSNDEDH	private_key_identifier private_KEK_key_identifier public_key_identifier output_KEK_key_identifier

Table 74. Callable services and parameters that trigger reference date processing (continued)

ICSF callable service	31-bit name	Parameter checked
EMV Scripting Service	CSNBESC	issuer_integrity_master_key_identifier issuer_confidentiality_master_key_identifier new_PIN_encrypting_key_identifier current_PIN_encrypting_key_identifier
EMV Transaction (ARQC/ARPC) Service	CSNBEAC	issuer_master_key_identifier issuer_ARPC_master_key_identifier
EMV Verification Functions	CSNBEVF	key_identifier
Encipher	CSNBENC	key_identifier
Encrypted PIN Generate	CSNBEPG	PIN_generating_key_identifier outbound_PIN_encrypting_key_identifier
Encrypted PIN Translate	CSNBPTR	input_PIN_encrypting_key_identifier output_PIN_encrypting_key_identifier
Encrypted PIN Verify	CSNBPVR	input_PIN_encrypting_key_identifier PIN_verifying_key_identifier
FPE Decipher	CSNBFPED	key_identifier
FPE Encipher	CSNBFPEE	key_identifier
FPE Translate	CSNBFPET	input_key_identifier output_key_identifier
Field Level Decipher	CSNBFLD	key_identifier
Field Level Encipher	CSNBFLE	key_identifier
Generate Issuer MK	CSNBGIM	transport_key_identifier
HMAC Generate	CSNBHMG	key_identifier
HMAC Verify	CSNBHMV	key_identifier
Key Data Set Metadata Write	CSFKDMW	label_list (only when "Last reference date" is being altered)
Key Export	CSNBKEX	source_key_identifier exporter_key_identifier
Key Generate	CSNBKGN	KEK_key_identifier_1 KEK_key_identifier_2
Key Generate2	CSNBKGN2	key_encrypting_key_identifier_1 key_encrypting_key_identifier_2
Key Import	CSNBKIM	importer_key_identifier
Key Test Extended	CSNBKYTX	KEK_key_identifier
Key Test2	CSNBKYT2	key_encrypting_key_identifier
Key Translate	CSNBKTR	input_KEK_key_identifier output_KEK_key_identifier
Key Translate2	CSNBKTR2	input_KEK_identifier output_KEK_identifier
MAC Generate	CSNBMGN	key_identifier
MAC Generate2	CSNBMGN2	key_identifier
MAC Verify	CSNBMVR	key_identifier
MAC Verify2	CSNBMVR2	key_identifier

Table 74. Callable services and parameters that trigger reference date processing (continued)

ICSF callable service	31-bit name	Parameter checked
Multiple Secure Key Import	CSNBSKM	key_encrypting_key_identifier
PIN Change/Unblock	CSNBPCU	authentication_issuer_master_key_identifier encryption_issuer_master_key_identifier new_reference_PIN_key_identifier current_reference_PIN_key_identifier
PKA Decrypt	CSNDPKD	key_identifier
PKA Encrypt	CSNDPKE	keyvalue PKA_key_identifier
PKA Key Generate	CSNDPKG	skeleton_key_identifier transport_key_identifier
PKA Key Import	CSNDPKI	source_key_identifier importer_key_identifier
PKA Key Translate	CSNDPKT	source_key_identifier source_transport_key_identifier target_transport_key_identifier
PKA Public Key Extract	CSNDPKX	source_key_identifier
PKCS #11 Derive Key	CSFPDVK	base_key_handle parms_list (Key handle of additional key)
PKCS #11 Derive Multiple Keys	CSFPDMK	base_key_handle parms_list (Key handle of additional key)
PKCS #11 Generate Keyed MAC	CSFPHMG	key_handle
PKCS #11 Get Attribute Value	CSFPGAV	handle
PKCS #11 Private Key Sign	CSFPPKS	key_handle
PKCS #11 Pseudo-Random Function	CSFPPRF	handle
PKCS #11 Public Key Verify	CSFPPKV	key_handle
PKCS #11 Token Record Create	CSFPTRC	handle
PKCS #11 Unwrap Key	CSFPUWK	unwrapping_key_handle initialization_vector
PKCS #11 Verify Keyed MAC	CSFPHMV	key_handle
PKCS #11 Wrap Key	CSFPWPK	source_key_handle wrapping_key_handle initialization_vector
PKCS #11 Secret Key Decrypt	CSFPSKD	key_handle
PKCS #11 Secret Key Encrypt	CSFPSKE	key_handle

Table 74. Callable services and parameters that trigger reference date processing (continued)

ICSF callable service	31-bit name	Parameter checked
PKDS Key Record Read	CSNDKRR	label
PKDS Key Record Read2	CSNDKRR2	label
Prohibit Export Extended	CSNBPEXX	KEK_key_identifier
Recover PIN From Offset	CSNBPFO	PIN_encryption_key_identifier PIN_generation_key_identifier
Remote Key Export	CSNDRKX	trusted_block_identifier transport_key_identifier importer_key_identifier source_key_identifier
Restrict Key Attribute	CSNBRKA	key_encrypting_key_identifier
Secure Key Import	CSNBSKI	importer_key_identifier
Secure Key Import2	CSNBSKI2	key_encrypting_key_identifier
Secure Messaging for Keys	CSNBSKY	input_key_identifier key_encrypting_key_identifier secmsg_key_identifier
Secure Messaging for PINs	CSNBSPN	PIN_encrypting_key_identifier secmsg_key_identifier
SET Block Compose	CSNDSBC	RSA_public_key_identifier
SET Block Decompose	CSNDSBD	RSA_private_key_identifier DES_key_block (second 64 bytes)
Symmetric Algorithm Decipher	CSNBSAD	key_identifier
Symmetric Algorithm Encipher	CSNBSAE	key_identifier
Symmetric Key Decipher	CSNBSYD	key_identifier
Symmetric Key Encipher	CSNBSYE	key_identifier
Symmetric Key Export	CSNDSYX	source_key_identifier transporter_key_identifier
Symmetric Key Export with Data	CSNDSXD	source_key_identifier RSA_public_key_identifier
Symmetric Key Generate	CSNDSYG	key_encrypting_key_identifier RSA_public_key_identifier
Symmetric Key Import	CSNDSYI	RSA_private_key_identifier
Symmetric Key Import2	CSNDSYI2	transport_key_identifier
Symmetric MAC Generate	CSNBSMG	key_identifier
Symmetric MAC Verify	CSNBSMV	key_identifier

Table 74. Callable services and parameters that trigger reference date processing (continued)

ICSF callable service	31-bit name	Parameter checked
Transaction Validation	CSNBTRV	transaction_key_identifier
Trusted Block Create	CSNDTBC	input_block_identifier transport_key_identifier
TR-31 Export	CSNBT31X	source_key_identifier unwrap_kek_identifier wrap_kek_identifier
TR-31 Import	CSNBT31I	unwrap_kek_identifier wrap_kek_identifier
Unique Key Derive	CSNBUKD	base_derivation_key_identifier transport_key_identifier
VISA CVV Service Generate	CSNBCSG	CVV_key_A_Identifier CVV_key_B_Identifier
VISA CVV Service Verify	CSNBCSV	CVV_key_A_Identifier CVV_key_B_Identifier

Appendix H. Questionable (Weak) Keys

If any of the eight-byte parts of the new master-key compares equal to one of the weak DES-keys, the service fails.

These are considered questionable DES keys:

```
01 01 01 01 01 01 01 01 / weak /
FE FE FE FE FE FE FE FE / weak /
1F 1F 1F 1F 0E 0E 0E 0E / weak /
E0 E0 E0 E0 F1 F1 F1 F1 / weak /
01 FE 01 FE 01 FE 01 FE /semi-weak /
FE 01 FE 01 FE 01 FE 01 /semi-weak /
1F E0 1F E0 0E F1 0E F1 /semi-weak /
E0 1F E0 1F F1 0E F1 0E /semi-weak /
01 E0 01 E0 01 F1 01 F1 /semi-weak /
E0 01 E0 01 F1 01 F1 01 /semi-weak /
1F FE 1F FE 0E FE 0E FE /semi-weak /
FE 1F FE 1F FE 0E FE 0E /semi-weak /
01 1F 01 1F 01 0E 01 0E /semi-weak /
1F 01 1F 01 0E 01 0E 01 /semi-weak /
E0 FE E0 FE F1 FE F1 FE /semi-weak /
FE E0 FE E0 FE F1 FE F1 /semi-weak /
1F 1F 01 01 0E 0E 01 01 /possibly semi-weak /
01 1F 1F 01 01 0E 0E 01 /possibly semi-weak /
1F 01 01 1F 0E 01 01 0E /possibly semi-weak /
01 01 1F 1F 01 01 0E 0E /possibly semi-weak /
E0 E0 01 01 F1 F1 01 01 /possibly semi-weak /
FE FE 01 01 FE FE 01 01 /possibly semi-weak /
FE E0 1F 01 FE F1 0E 01 /possibly semi-weak /
E0 FE 1F 01 F1 FE 0E 01 /possibly semi-weak /
FE E0 01 1F FE F1 01 0E /possibly semi-weak /
E0 FE 01 1F F1 FE 01 0E /possibly semi-weak /
E0 E0 1F 1F F1 F1 0E 0E /possibly semi-weak /
FE FE 1F 1F FE FE 0E 0E /possibly semi-weak /
FE 1F E0 01 FE 0E F1 01 /possibly semi-weak /
E0 1F FE 01 F1 0E FE 01 /possibly semi-weak /
FE 01 E0 1F FE 01 F1 0E /possibly semi-weak /
E0 01 FE 1F F1 01 FE 0E /possibly semi-weak /
01 E0 E0 01 01 F1 F1 01 /possibly semi-weak /
1F FE E0 01 0E FE F1 01 /possibly semi-weak /
1F E0 FE 01 0E F1 FE 01 /possibly semi-weak /
01 FE FE 01 01 FE FE 01 /possibly semi-weak /
1F E0 E0 1F 0E F1 F1 0E /possibly semi-weak /
01 FE E0 1F 01 FE F1 0E /possibly semi-weak /
01 E0 FE 1F 01 F1 FE 0E /possibly semi-weak /
1F FE FE 1F 0E FE FE 0E /possibly semi-weak /
E0 01 01 E0 F1 01 01 F1 /possibly semi-weak /
FE 1F 01 E0 FE 0E 10 F1 /possibly semi-weak /
FE 01 1F E0 FE 01 0E F1 /possibly semi-weak /
E0 1F 1F E0 F1 0E 0E F1 /possibly semi-weak /
FE 01 01 FE FE 01 01 FE /possibly semi-weak /
E0 1F 01 FE F1 0E 01 FE /possibly semi-weak /
E0 01 1F FE F1 01 0E FE /possibly semi-weak /
FE 1F 1F FE FE 0E 0E FE /possibly semi-weak /
1F FE 01 E0 E0 FE 01 F1 /possibly semi-weak /
01 FE 1F E0 01 FE 0E F1 /possibly semi-weak /
1F E0 01 FE 0E F1 01 FE /possibly semi-weak /
01 E0 1F FE 01 F1 0E FE /possibly semi-weak /
01 01 E0 E0 01 01 F1 F1 /possibly semi-weak /
1F 1F E0 E0 0E 0E F1 F1 /possibly semi-weak /
1F 01 FE E0 0E 01 FE F1 /possibly semi-weak /
01 1F FE E0 01 0E FE F1 /possibly semi-weak /
```

```
1F 01 E0 FE 0E 01 F1 FE /possibly semi-weak /
01 1F E0 FE 01 E0 F1 FE /possibly semi-weak /
01 01 FE FE 01 01 FE FE /possibly semi-weak /
1F 1F FE FE 0E 0E FE FE /possibly semi-weak /
FE FE E0 E0 FE FE F1 F1 /possibly semi-weak /
E0 FE FE E0 F1 FE FE F1 /possibly semi-weak /
FE E0 E0 FE FE F1 F1 FE /possibly semi-weak /
E0 E0 FE FE F1 F1 FE FE /possibly semi-weak /
```

Appendix I. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact z/OS or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, `3` or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE

keyword, but do not specify an option, the default option `KEEP` is applied. A default option also applies to the next higher dotted decimal number. In this example, if the `FILE` keyword is omitted, the default `FILE(KEEP)` is used. However, if you hear the lines `2? FILE`, `2.1`, `2.1.1! (KEEP)`, and `2.1.1 (DELETE)`, the default option `KEEP` applies only to the next higher dotted decimal number, `2.1` (which does not have an associated keyword), and does not apply to `2? FILE`. Nothing is used if the keyword `FILE` is omitted.

`* indicates an optional syntax element that is repeatable`
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line `5.1* data area`, you know that you can include one data area, more than one data area, or no data area. If you hear the lines `3* , 3 HOST, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The `*` symbol is equivalent to a loopback line in a railroad syntax diagram.

`+ indicates a syntax element that must be included`
The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST, and 2 STATE`, you know that you must include `HOST, STATE, or` both. Similar to the `*` symbol, the `+` symbol can repeat a particular item if it is the only item with that dotted decimal number. The `+` symbol, like the `*` symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

Terms and conditions for product documentation

Applicability

Personal use

Commercial use

Rights

NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: http://www.ibm.com/software/support/systemsz/lifecycle/

- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

Index

Printed in USA

SC14-7506-06