

z/OS
Cryptographic Services
Integrated Cryptographic Service Facility



Administrator's Guide

z/OS
Cryptographic Services
Integrated Cryptographic Service Facility



Administrator's Guide

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 385.

Fifteenth Edition, November 2009

This is a complete revision of SA22-7521-13.

This edition applies to Version 1 Release 11 of z/OS (5694–A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

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

International Business Machines Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405

FAX (Other Countries):

Your International Access Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)

Internet e-mail: mhvrdfs@us.ibm.com

World Wide Web: <http://www.ibm.com/systems/z/os/zos/webqs.html>

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

Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

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

© Copyright International Business Machines Corporation 1997, 2009.

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

Contents

Figures	ix
Tables	xv
About this information	xvii
ICSF Features	xvii
Who should use this information	xviii
How to use this information	xviii
Where to find more information	xix
Related Publications	xx
Do You Have Problems, Comments, or Suggestions?	xx
Summary of Changes	xxiii
Chapter 1. Introduction	1
The Tasks of a Data Security System	1
The Role of Cryptography in Data Security	2
Symmetric Cryptography	2
Asymmetric Algorithm or Public Key Cryptography	3
Cryptographic Hardware Features supported by z/OS ICSF	4
Managing Crypto Express2 Features on an IBM System z9 EC, z9 BC, z10 EC and z10 BC	6
Managing Crypto Express3 Features on an IBM System z10 EC and z10 BC	6
Strength of Hardware Cryptography	7
The Role of Key Secrecy in Data Security	7
Chapter 2. Understanding Cryptographic Keys	9
Values of Keys	9
Types of Keys	9
Master Keys	10
Data-Encrypting Keys	10
Data-Translation Keys	11
MAC Keys	11
PIN Keys	12
Cryptographic Variable Keys	13
Transport Keys	13
Key Generating Keys	14
PKA Keys	14
Protection and control of cryptographic keys	15
Master Key Concept	15
Key Separation	15
Migrating from PCF Key Types	18
Protection of Distributed Keys	19
Protecting Keys Stored with a File	19
Remote key loading	20
Using DES Transport Keys to Protect Keys Sent between Systems	20
Using RSA Public Keys to Protect Keys Sent between Systems	21
Protection of Data	22
Triple DES for Privacy	24
Advanced Encryption Standard (AES)	24
Chapter 3. Managing Cryptographic Keys	25
Generating Cryptographic Keys	25

Enhanced key management for crypto assist instructions	25
TKDS key protection	26
Generating PKA Keys	26
Key Generator Utility Program (KGUP)	26
Key Generate Callable Service	27
Entering Keys	27
Entering master keys	28
Entering system keys into the cryptographic key data set (CKDS)	29
Entering keys into the cryptographic key data set (CKDS)	30
Entering keys into the PKDS	32
Entering cryptographic objects into the TKDS	32
Maintaining cryptographic keys	33
Setting up and maintaining the cryptographic key data set (CKDS)	33
Setting up and maintaining the PKDS	36
Distributing Cryptographic Keys	36
Common Cryptographic Architecture Key Distribution	36
ANSI X9.17 Key Distribution	40
Public Key Cryptographic Standard Key Distribution	41
Controlling PCICC, PCIXCC, CEX2C, and CEX3C services	41
Chapter 4. Using RACF to Protect Keys and Services	43
Steps for RACF-protecting keys and services.	43
Setting up profiles in the CSFKEYS general resource class	45
Setting up profiles in the CSFSERV general resource class	46
Defining a key store policy	50
Enabling access authority checking for key tokens	53
Enabling duplicate key label checking	56
Increasing the level of authority needed to modify key labels	57
Increasing the level of authority required to export symmetric keys	59
Controlling how cryptographic keys can be used	61
Enabling use of encrypted keys in Symmetric Key Encipher and Symmetric Key Decipher callable services	71
Chapter 5. Using the Pass Phrase Initialization Utility	73
Steps required when running the Pass Phrase Initialization Utility	73
SAF Protection	73
Running the Pass Phrase Initialization Utility	74
Steps for running PPINIT on a CCF system	74
Steps for running PPINIT on a PCIXCC, CEX2C, or CEX3C system	77
Steps for running PPINIT with AES master key support	79
Steps for adding a PCICC after first time Pass Phrase Initialization.	82
Steps for adding a PCIXCC, CEX2C, or CEX3C after first time Pass Phrase Initialization	84
Migrating to a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC server.	87
PPINIT Recovery	88
Steps recovering with a CCF (with or without a PCICC)	88
Steps recovering with a PCIXCC, CEX2C, or CEX3C.	90
Initializing multiple systems with pass phrase initialization utility	91
Chapter 6. Managing Master Keys - CCF and PCICC	93
Entering master key parts	93
Generating master key data for master key entry	94
Steps for entering the first master key part	101
Steps for entering intermediate key parts	104
Steps for entering the final key part	105
Steps for restarting the key entry process	108

Initializing the CKDS and PKDS at First-Time Startup	110
CKDS	110
PKDS	113
Refreshing the CKDS at any time	115
Refreshing the PKDS at any time.	116
Reentering master keys when they have been cleared	117
Steps to reenter cleared master keys	117
Steps for changing master keys	119
DES master keys and the CKDS	119
PKA master keys and the PKDS	124
Steps for enabling and disabling PKA services	124
Steps for changing PKA master keys	125
Steps for reenciphering and refreshing the PKDS.	129
Steps for setting the SMK equal to the KMMK	131
Steps for clearing master keys.	132
Steps for adding a PCICC after CCF initialization.	132
Chapter 7. Managing Master Keys - PCIXCC, CEX2C, or CEX3C	135
Entering master key parts	135
Generating master key data for master key entry	135
Steps for entering the first master key part	142
Steps for entering intermediate key parts	145
Steps for entering the final key part	147
Steps for restarting the key entry process	150
Initializing the CKDS and PKDS at First-Time Startup	152
CKDS.	152
PKDS.	158
Refreshing the CKDS at Any Time	160
Refreshing the PKDS at any time	161
Reentering master keys when they have been cleared	162
Steps for changing master keys	163
Symmetric Master Keys and the CKDS	164
Asymmetric master keys and the PKDS	167
Steps for enabling and disabling PKA services	168
Steps for changing asymmetric master keys.	169
Steps for reenciphering and refreshing the PKDS.	172
Steps for clearing master keys.	174
Steps for adding PCIXCC, CEX2C, or CEX3C coprocessors after initialization	174
Chapter 8. Key Management on Systems without Coprocessors.	177
Initializing the CKDS at First-Time Startup	177
Steps for initializing a CKDS	177
Refreshing the CKDS at Any Time	178
Callable services.	180
Chapter 9. Running in a Sysplex Environment.	181
CKDS management in a sysplex	181
Setting DES and AES Master Keys when Sharing a CKDS	181
Changing DES and AES Master Keys when Sharing a CKDS	182
PKDS management in a sysplex	183
Steps for changing PKA master keys when sharing a PKDS.	183
Steps for refreshing the PKDS.	184
Sharing and migrating a CKDS/PKDS between a CCF system and a PCIXCC, CEX2C, or CEX3C system	185
CCF only system	186

CCF with PCICCs	187
TKDS management in a sysplex	189

Chapter 10. Managing Cryptographic Keys by Using the Key Generator

Utility Program	191
Steps for disallowing dynamic CKDS updates during KGUP updates.	192
Using KGUP for key exchange	194
Using KGUP control statements	196
General Rules for CKDS Records	196
Syntax of the ADD and UPDATE Control Statements	197
Using the ADD and UPDATE control statements for key management and distribution functions	203
Syntax of the RENAME Control Statement	209
Syntax of the DELETE Control Statement	209
Syntax of the SET Control Statement	210
Syntax of the OPKYLOAD Control Statement	211
Examples of Control Statements	211
Specifying KGUP data sets	218
Submitting a job stream for KGUP	223
Enabling Special Secure Mode	224
Running KGUP Using the MVS/ESA Batch Local Shared Resource (LSR) Facility	224
Reducing Control Area Splits and Control Interval Splits from a KGUP Run	225
Refreshing the In-Storage CKDS	225
Using KGUP Panels	226
Steps for creating KGUP control statements using the ICSF panels	227
Steps for specifying data sets using the ICSF panels	243
Steps for creating the job stream using the ICSF panels	246
Steps for refreshing the current CKDS using the ICSF panels	250
Scenario of Two ICSF Systems Establishing Initial Transport Keys	251
Scenario of an ICSF System and a PCF System Establishing Initial Transport Keys	253
Scenario of an ICSF System and 4758 PCI Cryptographic Coprocessor Establishing Initial Transport Keys	255

Chapter 11. Viewing and Changing System Status 257

Displaying administrative control functions	257
Displaying coprocessor or accelerator status - CCF, PCICC, PCICA	259
Displaying coprocessor or accelerator status - PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A	261
Changing coprocessor or accelerator status - CCF, PCICC, and PCICA	264
Changing coprocessor or accelerator status - PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A	264
Deactivating the last coprocessor.	265
Displaying coprocessor hardware status - CCF and PCICC	266
Displaying coprocessor hardware status - PCIXCC, CEX2C, and CEX3C	273
Displaying installation options	280
Displaying PCICC default roles	287
Displaying PCIXCC, CEX2C, and CEX3C default roles.	289
Displaying installation exits	293
Displaying installation-defined callable services	301

Chapter 12. Managing User Defined Extensions 305

Display UDXs for a coprocessor	306
Display coprocessors for a UDX	307
Authorize a UDX.	308

Chapter 13. Using the Utility Panels to Encode and Decode Data	309
Steps for encoding data	309
Steps for decoding data	310
Chapter 14. Using the Utility Panels to Manage Keys in the PKDS	313
RACF Protecting ICSF Services used by the New Panels.	313
Generate a new RSA public/private PKDS key pair record	315
Delete an existing key record	315
Export a public key to an X.509 certificate for importation elsewhere.	315
Import a public key from an X.509 certificate received from elsewhere	316
Processing Indicators	316
Success	316
Failure	317
Chapter 15. Using PKCS11 Token Browser Utility Panels	321
RACF Protecting ICSF Services used by the Token Browser Utility Panels	321
Token browser panel utility	323
Token Browser main panel	323
Token Create Successful.	324
Token Delete Confirmation	324
Token Delete Successful.	324
Object Delete Successful.	325
List Token panel	325
Token Details panel.	325
Data Object Details panel	327
Certificate Object Details panel	327
Secret Key Object Details panel	328
Public Key Object Details panel	329
Private Key Object Details panel	332
Domain Parameters Object Details panel.	336
Chapter 16. Using the ICSF Utility Program CSFEUTIL	339
Reenciphering a disk copy of a CKDS and changing the master key.	339
Refreshing the in-storage CKDS using a utility program	341
Loading DES and PKA master keys using a pass phrase	341
Return and reason codes for the CSFEUTIL program	342
CSFWEUTL	344
Chapter 17. Using the ICSF Utility Program CSFPUTIL	349
Initializing a PKDS	349
Reenciphering a PKDS	350
Refreshing the in-storage copy of the PKDS	350
Return and reason codes for the CSFPUTIL program	351
CSFWPUTL	352
Chapter 18. Using the ICSF Utility Program CSFDUTIL	357
Using the Duplicate Token Utility	357
CSFDUTIL output	357
Return and reason codes for the CSFDUTIL program	358
CSFWDUTL	359
Appendix A. CCC Bit Assignments	361
Appendix B. Control Vector Table.	363
Appendix C. Supporting Algorithms and Calculations	365

Checksum Algorithm	365
Algorithm for calculating a verification pattern	367
AES master key verification pattern algorithm	367
Algorithm for calculating an authentication pattern	367
Pass Phrase Initialization master key calculations	368
The MDC-4 Algorithm for Generating Hash Patterns	368
Notations Used in Calculations	368
MDC-1 Calculation	369
MDC-4 Calculation	369
Appendix D. PR/SM Considerations during Key Entry	371
Allocating Cryptographic Resources to a Logical Partition	371
Allocating Resources on z/990 or z890	371
Allocating Resources on CCF Systems	372
Entering the Master Key or Other Keys in LPAR Mode	373
Reusing or Reassigning a Domain	373
Appendix E. Callable services affected by key store policy	375
Summary of Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions	379
Appendix F. Questionable (Weak) Keys	381
Appendix G. Accessibility	383
Using assistive technologies	383
Keyboard navigation of the user interface.	383
z/OS information	383
Notices	385
Programming Interface Information	386
Trademarks.	386
Index	389

Figures

1. The z/OS ICSF Library	xxi
2. Keys Protected in a System	17
3. Keys Protected in a File Outside the System	19
4. Keys and PINs Protected When Sent between Two Systems	21
5. Distributing a DES Data-Encrypting Key Using an RSA Cryptographic Scheme	22
6. Data Protected When Sent between Intermediate Systems	23
7. Updating the In-Storage Copy and the Disk Copy of the CKDS	35
8. Key Sent from System A to System B	38
9. Keys Sent between System A and System B	39
10. ANSI X9.17 Keys Sent between System A and System B	40
11. Selecting the Pass Phrase Initialization Option on the ICSF Primary Menu Panel	74
12. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel	75
13. Entering Options on the Pass Phrase MK/KDS Initialization Panel	76
14. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel	77
15. Entering Options on the Pass Phrase MK/CKDS/PKDS Initialization Panel	78
16. Pass Phrase MK/CKDS/PKDS Initialization Panel	79
17. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel	80
18. Entering Options on the Pass Phrase MK/CKDS/PKDS Initialization Panel	81
19. Pass Phrase MK/CKDS/PKDS Initialization Panel	82
20. Selecting the Pass Phrase Initialization Option on the ICSF Primary Menu Panel	83
21. ICSF Pass Phrase MK/KDS Initialization Panel	83
22. Entering Options on the Pass Phrase MK/KDS Initialization Panel	84
23. Selecting the Pass Phrase Initialization Option on the ICSF Primary Menu Panel	85
24. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel	86
25. Entering Options on the Pass Phrase MK/CKDS/PKDS Initialization Panel	87
26. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel	88
27. Coprocessor Hardware Status Panel	90
28. Selecting the Utility Option on the ICSF Primary Menu Panel	96
29. ICSF Utilities Panel	96
30. ICSF Random Number Generator Panel	97
31. ICSF Random Number Generator Panel with Generated Numbers	97
32. Selecting the Checksum Option on the ICSF Utilities Panel	98
33. ICSF Checksum and Verification and Hash Pattern Panel	99
34. Key Type Selection Panel Displayed During Hardware Key Entry	99
35. ICSF Checksum and Verification Pattern Panel	100
36. Checksum, Verification Pattern, and Hash Pattern Calculated for a DES Master Key Part	100
37. Selecting the Coprocessor Management option on the primary menu panel	101
38. Selecting the coprocessor on the Coprocessor Management Panel	102
39. Master Key Entry Panel	102
40. The Master Key Entry Panel Following Key Part Entry	103
41. The Master Key Entry Panel for Intermediate Key Values	104
42. The Master Key Entry Panel with Intermediate Key Values	105
43. The Master Key Entry Panel when entering Final Key Values	106
44. The Master Key Entry Panel with Final Key Values	107
45. Selecting Reset on the Master Key Entry Panel	108
46. Confirm Restart Request Panel	109
47. The Master Key Entry Panel Following Reset Request	109
48. Selecting the Master Key option on the primary menu panel	111
49. ICSF Master Key Management Panel	112
50. ICSF Initialize a CKDS Panel	112
51. Selecting the Master Key option on the primary menu panel	114
52. ICSF Master Key Management Panel	114
53. ICSF Initialize a PKDS Panel	115

54. Refresh PKDS	115
55. Selecting the Refresh Option on the ICSF Initialize a CKDS Panel	116
56. Selecting the Set Host Master Key Option on the ICSF Master Key Management Panel	118
57. Selecting the Reencipher CKDS option on the ICSF Master Key Management Panel	121
58. Reencipher CKDS	122
59. Change Master Key Panel	123
60. Selecting Administrative Control on the ICSF Primary Menu Panel	124
61. Enabling and Disabling the PKA Callable Services	125
62. Selecting the coprocessor on the Coprocessor Management Panel	125
63. The Master Key Entry Panel to Reset Registers	126
64. Confirm Restart Request Panel	126
65. The Master Key Entry Panel with First Key Values	127
66. The Master Key Entry Panel with Final Key Values	128
67. Selecting the Reencipher PKDS Option on the Master Key Management Panel	129
68. Reencipher PKDS	130
69. Selecting the Activate PKDS Option on the Master Key Management Panel	130
70. Refresh PKDS	130
71. ICSF Utilities Panel	131
72. ICSF Master Key Values from Pass Phrase Panel	131
73. Selecting a coprocessor on the Coprocessor Management Panel	133
74. The Master Key Entry Panel to Reset Registers	133
75. Selecting the Utility Option on the ICSF Primary Menu Panel	137
76. ICSF Utilities Panel	138
77. ICSF Random Number Generator Panel	138
78. ICSF Random Number Generator Panel with Generated Numbers	138
79. Selecting the Checksum Option on the ICSF Utilities Panel	139
80. ICSF Checksum and Verification and Hash Pattern Panel	140
81. Key Type Selection Panel Displayed During Hardware Key Entry	141
82. ICSF Checksum and Verification Pattern Panel	141
83. Selecting the Coprocessor Management option on the primary menu panel	142
84. Selecting the coprocessor on the Coprocessor Management Panel	143
85. Master Key Entry Panel	143
86. Master Key Entry Panel	144
87. The Master Key Entry Panel Following Key Part Entry	145
88. The Master Key Entry Panel for Intermediate Key Values	146
89. The Master Key Entry Panel with Intermediate Key Values	147
90. The Master Key Entry Panel when entering Final Key Values	148
91. The Master Key Entry Panel with Final Key Values	149
92. Selecting Reset on the Master Key Entry Panel	151
93. Confirm Restart Request Panel	151
94. The Master Key Entry Panel Following Reset Request	152
95. Selecting the Master Key option on the primary menu panel	153
96. ICSF Master Key Management Panel	154
97. ICSF Initialize a CKDS Panel	154
98. ICSF Initialize a CKDS Panel if AES master keys are supported	154
99. Selecting the Master Key option on the primary menu panel	156
100. ICSF Master Key Management Panel	156
101. ICSF Initialize a CKDS Panel if AES master keys are supported	157
102. ICSF Initialize a CKDS Panel	157
103. ICSF Master Key Management Panel	158
104. Selecting the Master Key option on the primary menu panel	159
105. ICSF Master Key Management Panel	159
106. ICSF Initialize a PKDS Panel	159
107. Refresh PKDS	160
108. Selecting the Refresh Option on the ICSF Initialize a CKDS Panel	161
109. Selecting the Set Host Master Key Option on the ICSF Master Key Management Panel	163

110. Selecting the Reencipher CKDS option on the ICSF Master Key Management Panel	166
111. Reencipher CKDS	166
112. Change Master Key Panel	167
113. Selecting Administrative Control on the ICSF Primary Menu Panel	168
114. Enabling and Disabling the PKA Callable Services	168
115. Selecting the coprocessor on the Coprocessor Management Panel	169
116. The Master Key Entry Panel with First Key Values	170
117. The Master Key Entry Panel with Final Key Values	171
118. Selecting the Reencipher PKDS Option on the Master Key Management Panel	172
119. Reencipher PKDS	173
120. Selecting the Refresh PKDS Option on the Master Key Management Panel.	173
121. Refresh PKDS	173
122. Selecting a coprocessor on the Coprocessor Management Panel	174
123. The Master Key Entry Panel to Reset Registers	175
124. ICSF Master Key Management Panel.	178
125. ICSF Initialize a CKDS Panel.	178
126. ICSF Master Key Management Panel.	179
127. ICSF Initialize a CKDS Panel.	179
128. Administrative Control Functions.	184
129. ICSF Master Key Management Panel.	185
130. Selecting the Administrative Control Option on the Primary Menu Panel	193
131. Selecting to Disallow Dynamic CKDS Access on User Control Functions Panel	193
132. ADD and UPDATE Control Statement Syntax	198
133. RENAME Control Statement Syntax	209
134. DELETE Control Statement Syntax	209
135. SET Control Statement Syntax	210
136. OPKYLOAD Control Statement Syntax	211
137. Diagnostics Data Set Example	221
138. KGUP Job Stream	223
139. Selecting the KGUP Option on the Primary Menu Panel	226
140. Key Administration Panel	227
141. Selecting the Create Option on the Key Administration Panel	227
142. KGUP Control Statement Data Set Specification Panel	228
143. Entering a Data Set Name on the KGUP Control Statement Data Set Specification Panel	229
144. Member Selection List Panel	230
145. Entering Data Set Information on the Allocation Panel.	230
146. KGUP Control Statement Menu Panel	231
147. Create ADD, UPDATE, or DELETE Key Statement Panel	232
148. Selecting the ADD Function on the Create ADD, UPDATE, or DELETE Key Statement Panel	233
149. Selecting a Key on the Key Type Selection Panel	234
150. Completing the Create ADD, UPDATE, or DELETE Key Statement Panel	235
151. Specifying Multiple Key Labels on the Group Label Panel	237
152. Create ADD, UPDATE, or DELETE Key Statement Panel Showing Successful Update.	238
153. Selecting the Rename Option on the KGUP Control Statement Menu Panel	238
154. Create RENAME Control Statement Panel	239
155. Selecting a Key Type on the Key Type Selection Panel	239
156. Completing the Create RENAME Control Statement Panel	240
157. Selecting the Set Option on the KGUP Control Statement Menu Panel	241
158. Create SET Control Statement Panel	241
159. Completing the Create SET Control Statement Panel	242
160. Selecting the Edit Option on the KGUP Control Statement Menu Panel	242
161. Edit Control Statement Initial Display Panel	243
162. Edit Control Statement Data Set with Insert	243
163. Selecting the Specify Data Set Option on the Key Administration Panel	244
164. Specify KGUP Data Sets Panel	244
165. Completing the Specify KGUP Data Sets Panel	246

166. Invoking KGUP by Selecting the Submit Option on the Key Administration Panel	246
167. Set KGUP JCL Job Card Panel	247
168. KGUP JCL Set for Editing and Submitting (Files Exist)	248
169. KGUP JCL Set for Editing and Submitting (Files Do Not Exist)	249
170. Selecting the Refresh Option on the Key Administration Panel.	250
171. Refresh In-Storage CKDS	251
172. Key Exchange Establishment between Two ICSF Systems	251
173. Key Exchange Establishment between an ICSF System and a PCF System	253
174. Key Exchange Establishment between a 4758 PCI Cryptographic Coprocessor System and an ICSF System.	255
175. Primary Panel	258
176. Administrative Control Functions Panel	258
177. Selecting Coprocessor Status on the Primary Menu Panel	259
178. Coprocessor Management Panel	260
179. Selecting for Coprocessor Status on the Primary Menu Panel	262
180. Coprocessor Management Panel	262
181. Coprocessor Management Panel	264
182. Coprocessor Management Panel	265
183. Coprocessor Management Panel	266
184. Selecting the coprocessor on the Coprocessor Management Panel	267
185. Coprocessor Hardware Status Panel	268
186. Selecting the coprocessor on the Coprocessor Management Panel	274
187. Coprocessor Hardware Status Panel	275
188. Selecting the Installation Options on the Primary Menu Panel	281
189. Installation Options Panel	281
190. Installation Options Display Panel	282
191. Selecting for Coprocessor Status on the Primary Menu Panel	287
192. Coprocessor Management Panel	287
I 193. Default Role Status Display Panel	288
I 194. Default Role Status Display Panel – part 2	289
195. Selecting for Coprocessor Status on the Primary Menu Panel	290
196. Coprocessor Management Panel	290
I 197. Default Role Status Displayed for a system without TKE connected.	291
I 198. Default Role Status Displayed for a system without TKE connected - part 2.	292
I 199. Default Role Status Displayed for a system without TKE connected – part 3	293
200. Selecting the Installation Options and Hardware Status Option on the Primary Menu Panel	294
201. Installation Options Panel	294
202. First Installation Exits Display Panel	295
203. Second Installation Exits Display Panel	296
204. Third Installation Exits Display Panel	297
205. Fourth Installation Exits Display Panel	298
206. Selecting the Installation Options and Hardware Status Option on the Primary Menu Panel	302
207. Installation Options Panel	302
208. Installation-Defined Services Display Panel.	302
209. Selecting the UDX MGMT Option on the ICSF Primary Menu Panel	305
210. User Defined Extensions Management Panel	306
211. Authorized UDX Coprocessor Selection Panel	306
212. Authorized UDXs Panel	307
213. Coprocessors for Authorized UDXs Panel	307
214. Coprocessors for Authorized UDXs Panel	307
215. Authorize UDXs Panel	308
216. Selecting the Utilities Option on the Primary Menu Panel	309
217. Selecting the Encode Option on the Utilities Panel	310
218. Encode Panel	310
219. Selecting the Decode Option on the Utilities Panel	311
220. Decode Panel	311

221. Selecting the PKDSKEYS option on the ICSF Utilities Panel	314
222. ICSF PKDS Keys Panel.	314
223. PKDS Key Request Successful	316
224. PKDS Public Key Export Successful	317
225. PKDS Public Key Import Successful	317
226. PKDS Key Request Failed	318
227. PKDS Public Key Export Failure.	318
228. PKDS Public Key Import Failure.	319
229. Selecting the PKCS11 TOKEN option on the ICSF Utilities Panel	323
230. ICSF Token Management - Main Menu Panel.	324
231. ICSF Token Management - PKCS11 Token Create Successful panel	324
232. ICSF Token Management - PKCS11 Token Delete Confirmation panel.	324
233. ICSF Token Management - PKCS11 Token Delete Successful panel	325
234. ICSF Token Management - PKCS11 Object Delete Successful panel	325
235. ICSF Token Management - List Token Panel	325
236. ICSF Token Management - Token Details panel	326
237. ICSF Token Management - Data Object Details panel.	327
238. ICSF Token Management - Certificate Object Details panel.	328
239. ICSF Token Management - Secret Key Object Details panel	329
240. ICSF Token Management - Public Key Object Details panel	330
241. ICSF Token Management - Private Key Object Details panel – Part 1	333
242. ICSF Token Management - Private Key Object Details panel – Part 2	334
I 243. ICSF Token Management - Domain Parameters Object Details panel	336
244. Addition Table	366
245. Shift Table	366
246. The Clear Master Key Entry Panel - CCF and PCICC.	374
I 247. The Clear Master Key Entry Panel - PCIXCC, CEX2C, and CEX3C	374

Tables

1.	PCF and Corresponding ICSF Key Types	18
2.	Methods for Entering Each Key Type into the CKDS	30
3.	Key Store Policy controls	51
4.	Key Store Policy controls: The Key Token Authorization Checking controls	54
5.	Key Store Policy controls: The Default Key Label Checking controls	56
6.	Key Store Policy controls: The Duplicate Key Token Checking controls	57
7.	Increased access authority required to modify key labels when Granular Key Label Access control is enabled	57
8.	Key Store Policy controls: The Granular Key Label Access controls	58
9.	Key Store Policy controls: The Symmetric Key Label Export controls	60
10.	Keyword settings for symmetric key export using the ICSF segment's SYMEXPORTABLE field	64
11.	Key Store Policy controls: The PKA Key Management Extensions controls	69
12.	Default and Optional OUTTYPES Allowed for Each Key TYPE	200
13.	Keyword Combinations Permitted in ADD and UPDATE Control Statements	203
14.	Data Set Name Options	228
15.	Selecting Range and Label Options	235
16.	Selecting the Transport Key Label and Clear Key Label Options	236
17.	General ICSF Exits and Exit Identifiers	298
18.	Callable Service and its Exit Identifier	298
19.	Compatibility Service and its Exit Identifier	301
20.	Token access levels	322
21.	Resources in the CSFSERV class for token services	322
22.	Information displayed in Public Key Object Details panel for RSA, DSA, Diffie-Hellman, and Elliptic Curve keys	331
23.	Information displayed in Private Key Object Details panel for RSA, DSA, Diffie-Hellman, and Elliptic Curve keys	335
24.	Information displayed in Domain Parameters Object Details panel for DSA and Diffie-Hellman domain parameters	337
25.	CKDS information from CSFDUTIL	357
26.	PKDS information from CSFDUTIL	358
27.	Default Control Vector Values	363
28.	Planning LPARs domain and cryptographic coprocessor	372
29.	Callable services and parameters affected by key store policy	375
30.	Callable services that are affected by the no duplicates key store policy controls	378
31.	Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions (label)	379
32.	Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions (token)	379

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 includes these components:

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

ICSF is a software 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.

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 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 Digital Signature Standard (DSS) public and private keys on a multi-user, multi-application platform.
- It provides the ability to generate RSA key pairs within the secure hardware boundary of the PCI Cryptographic Coprocessor, PCI X Cryptographic Coprocessor, Crypto Express2 Coprocessor, or Crypto Express3 Coprocessor.

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 three 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 9, 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 25, 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, “Using RACF to Protect Keys and Services,” on page 43, describes how you can use z/OS Security Server RACF to 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 5, “Using the Pass Phrase Initialization Utility,” on page 73 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 6, “Managing Master Keys - CCF and PCICC,” on page 93 describes how to enter, activate, and manage master keys with both the Cryptographic Coprocessor Feature and the PCI Cryptographic Coprocessor.
- Chapter 7, “Managing Master Keys - PCIXCC, CEX2C, or CEX3C,” on page 135 describes how to enter, activate, and manage master keys with the PCI X Cryptographic Coprocessor, Crypto Express2 Coprocessor, or Crypto Express3 Coprocessor.
- Chapter 8, “Key Management on Systems without Coprocessors,” on page 177, describes how to manage clear AES and DES DATA keys on a system that does not have any cryptographic coprocessors or accelerators.
- Chapter 9, “Running in a Sysplex Environment,” on page 181, describes various considerations for the PKDS and CKDS when running in a sysplex.

- Chapter 10, “Managing Cryptographic Keys by Using the Key Generator Utility Program,” on page 191, 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 11, “Viewing and Changing System Status,” on page 257, 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, PCI management status, installation exits, and installation-defined services.
- Chapter 12, “Managing User Defined Extensions,” on page 305, describes how to use panels to manage your own cryptographic callable service.
- Chapter 13, “Using the Utility Panels to Encode and Decode Data,” on page 309, describes how to use utility panels to encipher and decipher data with a key that is not enciphered.
- Chapter 14, “Using the Utility Panels to Manage Keys in the PKDS,” on page 313, describes how to use the new PKDSKEYS option on the ICSF utilities panel to provide PKDS key management capability.
- Chapter 15, “Using PKCS11 Token Browser Utility Panels,” on page 321, describes how to use the new PKCS11 TOKEN option on the ICSF utilities panel to provide TKDS key management capability.
- Chapter 16, “Using the ICSF Utility Program CSFEUTIL,” on page 339, describes how to use the CSFEUTIL utility program to change master keys and refresh or reencipher the CKDS.
- Chapter 17, “Using the ICSF Utility Program CSFPUTIL,” on page 349, describes how to use the CSFPUTIL utility program to reencipher, activate and refresh a PKDS.
- Appendix A, “CCC Bit Assignments,” on page 361, contains selected CCC (crypto configuration control) definitions.
- Appendix B, “Control Vector Table,” on page 363, contains a table of the control vector values that are associated with each key type.
- Appendix C, “Supporting Algorithms and Calculations,” on page 365, shows algorithms that are used to calculate checksums, verification patterns, and other values.
- Appendix D, “PR/SM Considerations during Key Entry,” on page 371, discusses additional considerations when running in PR/SM logical partition mode.
- Appendix F, “Questionable (Weak) Keys,” on page 381, gives examples of questionable keys.
- “Notices” on page 385, discusses notices, programming interface information and trademarks.

Where to find more information

This information is supported by other publications in the ICSF library and other system libraries. The ICSF library is shown in Figure 1 on page xxi.

These publications contain additional ICSF information:

- *z/OS MVS System Codes*, SA22-7626
This publication describes the 18F abend code ICSF issues.
- *z/OS MVS System Management Facilities (SMF)*, SA22-7630
This publication describes SMF record type 82, where ICSF records events.
- *z/OS MVS Initialization and Tuning Guide*, SA22-7591

- *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- *z/OS MVS Programming: Callable Services for High-Level Languages*, SA22-7613
- *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608
- *z/OS MVS Programming: Extended Addressability Guide*, SA22-7614
- *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*, SA22-7609
- *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*, SA22-7610
- *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*, SA22-7611
- *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*, SA22-7612
- *MVS Batch Local Shared Resources*, GC28-1469
- *z/OS DFSMSdfp Storage Administration*, SC26-7402
- *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394

Related Publications

- *Support Element Operations Guide*
- *zSeries PR/SM Planning Guide*
- *zSeries Hardware Configuration Manager User's Guide*
- *zSeries Hardware Management Console Operations*
- *IBM Security Architecture: Securing the Open Client/Server Distributed Enterprise*
- *VTAM in a Parallel Sysplex Environment*
- *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, John Wiley & Sons, Inc.

Information for the PCI Cryptographic Coprocessor is found on the web at <http://www.ibm.com/security/cryptocards/html/library.shtml>.

- *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual for IBM 4758 Models 002 and 023*
- *IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide for the IBM 4758 Models 002 and 023*
- *IBM 4758 PCI Cryptographic Coprocessor General Information*
- *IBM 4758 PCI Cryptographic Coprocessor Installation*

Do You Have Problems, Comments, or Suggestions?

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

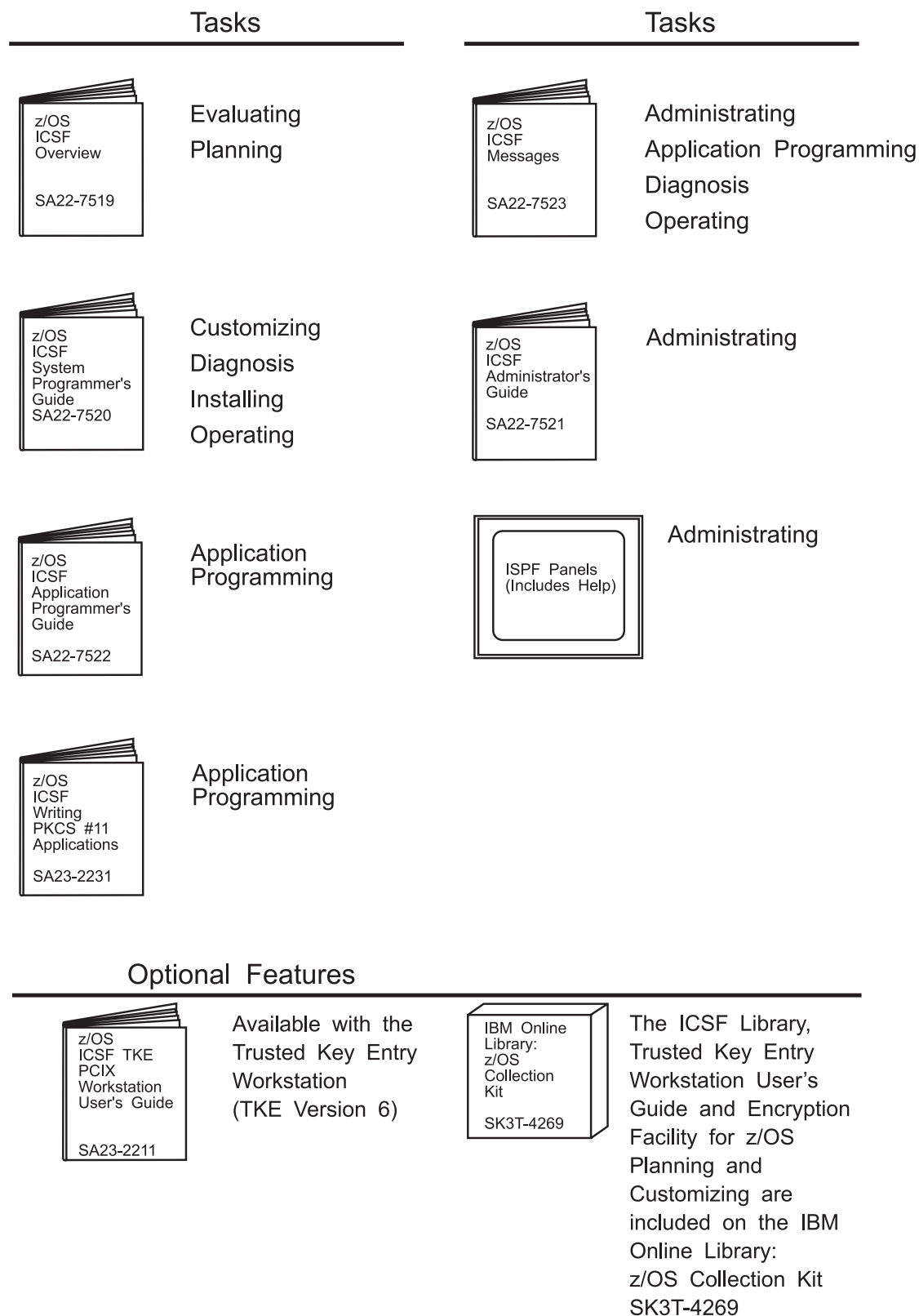


Figure 1. The z/OS ICSF Library

Summary of Changes

Summary of changes for SA22-7521-14 z/OS Version 1 Release 11

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SA22-7521-13, which supports z/OS Version 1 Release 10.

This document is for ICSF FMID HCR7770. This release of ICSF runs on z/OS V1R9, z/OS V1R10, and z/OS V1R11 and only on zSeries hardware.

New information

- Added information on the Crypto Express3 feature
- Added information on PKA Key Management Extensions control
- Added information on Symmetric Key Encipher and Symmetric Key Decipher support of encrypted data-encrypting keys in the CKDS.

Changed information

- Updated information on key store policy
- Updated PKCS11 token browser utility panels

Summary of changes for SA22-7521-13 z/OS Version 1 Release 10

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SA22-7521-12, which supports z/OS Version 1 Release 9.

This document is for ICSF FMID HCR7751. This release of ICSF runs on z/OS V1R7, z/OS V1R8, z/OS V1R9 and z/OS V1R10 and only on zSeries hardware.

New information

- Added support for secure key AES
 - Added new callable service CSNBSAD, CSNBSAD1, CSNESAD, and CSNESAD1 - Symmetric Algorithm Decipher
 - Added new callable service CSNBSAE, CSNBSAE1, CSNESAE, and CSNESAE1 - Symmetric Algorithm Encipher
- Added support for key store policy
 - new SMF 82 subtypes
- Additional SAF authorization checks when a key store policy is enabled
- Added a new utility to detect duplicate tokens in the CKDS or PKDS
- Added support for sysplex-wide consistency of PKDS
- Added support for an in-storage copy of the PKDS. This replaces the PKDS cache.
- Added support for IBM System z10 Business Class
- Added support for storing clear keys in the CKDS for a system without any cryptographic coprocessors
- Added support for PAN-14, PAN-15, PAN-17, PAN-18
 - Updated callable service CSNBCSG - VISA CVV Service Generate

- Updated callable service CSNBCSV - VISA CVV Service Verify
- Added support for IPv6
 - Added new callable service CSNBSMG, CSNBSMG1, CSNESMG and CSNESMG1 - Symmetric MAC Generate
 - Added new callable service CSNBSMV, CSNBSMV1, CSNESMV and CSNESMV1 - Symmetric MAC Verify
- Added new callable service CSFIQA and CSFIQA6 - ICSF Query Algorithm
- Added SAF class XCSFKEYS for Enhanced Keylabel Access Control

Changed information

- The symmetric-key master key is now called the DES master key
- These callable services have been changed to support secure key AES:
 - CSFIQF and CSFIQF6 - ICSF Query Function
 - CSNBKGN and CSNEKGN - Key Generate
 - CSNBKRC - Key Record Create
 - CSNBKRD - Key Record Delete
 - CSNBKRR - Key Record Read
 - CSNBKRW - Key Record Write
 - CSNBKYT - Key Test
 - CSNBKTB - Key Token Build
 - CSNBCKM and CSNECKM - Multiple Clear Key Import
 - CSNBSKM - Multiple Secure Key Import
 - CSNDSYX - Symmetric Key Export
 - CSNDSYG - Symmetric Key Generate
 - CSNDSYI - Symmetric Key Import
- KGUP changed to support secure key AES
- Panel changes for support of the AES master key
- Updated samples shipped in SYS1.SAMPLIB

Summary of changes for SA22-7521-12 z/OS Version 1 Release 9

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SA22-7521-11, which supports z/OS Version 1 Release 9.

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

New information

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

Changed information

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

- CPACF supports the AES algorithm for 192- and 256-bit keys for CSNBSYE, CSNBSYD, CSNBSYE1, CSNBSYD1

**Summary of changes
for SA22-7521-11
z/OS Version 1 Release 9**

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SA22-7521-10, which supports z/OS Version 1 Release 9.

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

New information

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

Changed information

- These callable services have been changed to support 4096-bit RSA keys:
 - CSNDPKD and CSNFPKD - PKA Decrypt
 - CSNDPKE and CSNFPKE - PKA Encrypt
 - CSNDPKX and CSNFPKX - PKA Key Extract
 - CSNDPKG and CSNFPKG - PKA Key Generate
 - CSNDPKI and CSNFPKI - PKA Key Import
 - CSNDPKB and CSNFPKB - PKA Key Token Build
 - CSNDKTC - PKA Key Token Change
 - CSNDKRC and CSNFKRC - PKDS Record Create
 - CSNDKRW - PKDS Record Write
 - CSNDKRD and CSNFKRD - PKDS Record Delete
 - CSNDRKX - Remote Key Export
 - CSNDSYX - Symmetric Key Export
 - CSNDSYG - Symmetric Key Generate
 - CSNDSYI - Symmetric Key Import
 - CSNDTBC - Trusted Block Create
 - CSNDDSG - Digital Signature Generate
 - CSNDDSV - Digital Signature Verify
- These callable services have been changed to support ISO-3 PIN block format:
 - CSNBCPE - Clear PIN Encrypt
 - CSNBEPG - Encrypted PIN Generate
 - CSNBPTR - Encrypted PIN Translate
 - CSNBPVR - Encrypted PIN Verify
 - CSNBPCU - PIN Change/Unblock
 - CSNBSPN - Secure Messaging for PINs
- The callable service, Random Number Generate Long (CSNBRNGL and CSNERNGL), now allows a user to specify the length when generating a random number.

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

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

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

Summary of changes for SA22-7521-10 z/OS Version 1 Release 9

This document contains information previously presented in *z/OS ICSF Administrator's Guide*, SA22-7521-09, which supports z/OS Version 1 Release 8.

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

New information

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

Changed information

- These callable services have been changed to support Cipher Feedback Mode (CFB) and PKCS #7 padding for encryption:
 - CSNBSYD - Symmetric Key Decipher (new CFB and PKCS-PAD keywords)
 - CSNBSYE - Symmetric Key Encipher (new CFB and PKCS-PAD keywords)
- Added SAF Protection to the steps required before running the PPINIT utility to indicate what services of CSFPMCI users need to be authorized to in order to use the utility.
- Added Initializing multiple systems with pass phrase initialization utility.
- Added to CP Assist for Cryptographic Functions (CPACF) that with only the CPACF feature installed, most of the ICSF panel options will not be available. When those options are selected, the message 'OPTION NOT ACTIVE' will be displayed in the upper right corner of the panel.
- Added improvements on auditing ICSF events.

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

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial

words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

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

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 several symmetric cryptography algorithms: The Data Encryption Algorithm, the Advanced Encryption Standard, and the Commercial Data Masking Facility.

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

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.

data. Someone could, of course, change both the data and the MAC. Therefore, 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 IBM System z10 Enterprise Class and IBM System z10 Business Class with the Nov. 2008 or later licensed internal code (LIC).

The Commercial Data Masking Facility

The Commercial Data Masking Facility (CDMF) defines a scrambling technique for data confidentiality. CDMF is a substitute for DES for those customers who have been previously prohibited from receiving IBM products that support DES data confidentiality services.

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

The CDMF data confidentiality algorithm is a cryptographic system that provides data masking and unmasking. The algorithm includes both a key-shortening process and a standard DES encryption and decryption process. The first process shortens the key to an effective length of 40 bits prior to its use in the data masking process. CDMF uses the DES algorithm with the shortened key to ensure confidence in the CDMF algorithm.

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)

- Digital Signature Standard (DSS)

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

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 *RSA's Frequently Asked Questions About Today's Cryptography* provide more information about the RSA public key algorithm.

The DSS Public Key Algorithm

The U.S. National Institute of Science and Technology (NIST) Digital Signature Standard (DSS) public key algorithm is based on the difficulty of the discrete logarithm problem. The discrete logarithm problem is to find x given a large prime p , a generator g and a value $y = (g^x) \bmod p$. In this equation, $^$ represents exponentiation. This problem is believed to be very hard when p is sufficiently large and x is a sufficiently large random number. For DSS, p is at least 512 bits, and x is 160 bits. DSS is defined in the NIST Federal Information Processing Standard (FIPS) 186 Digital Signature Standard.

A DSS key pair includes a private and a public key. The DSS private key is used to generate a digital signature, and the DSS public key is used to verify a digital signature.

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

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 Express3 Feature (CEX3C or CEX3A)

- available on IBM System z10 Enterprise Class and IBM System z10 Business Class
- 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

Crypto Express2-1P Feature

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

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

PCI X Cryptographic Coprocessor (PCIXCC)

- available on IBM @server zSeries 990 and IBM @server zSeries 890

CP Assist for Cryptographic Functions (CPACF)

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

- on the IBM @server zSeries 990 and IBM @server zSeries 890
 - SHA-1 algorithm is available
- 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 and IBM System z10 Business Class
 - 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, feature 3863, provides for clear key DES and TDES instructions.

If you want to include cryptographic hardware (CEX2C, CEX2A, CEX3C, CEX3A, PCIXCC, PCICA), 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 (SYM-MK and ASYM-MK)
2. Initialize the PKDS or CKDS.
3. Store keys in the PKDS or CKDS.

PCI Cryptographic Accelerator (PCICA)

The PCI Cryptographic Accelerator:

- available on IBM @server zSeries 800, IBM @server zSeries 900, IBM @server zSeries 990 and IBM @server zSeries 890
- provides support for clear keys in the CSNDPKD callable services for better performance. On z990 and z890, it also supports CSNDDSV and CSNDPKE.
- enables maximum SSL performance

Cryptographic Coprocessor Feature (CCF)

- available on IBM @server zSeries 800 and the IBM @server zSeries 900.

PCI Cryptographic Coprocessor

- available on IBM @server zSeries 800 and the IBM @server zSeries 900.

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 Crypto Express3 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. 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. If the support element hasn't 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 and z10 BC

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 Crypto 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. 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. If the support element hasn't 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 encode and decode callable services (CSNBECO and CSNBDCO) 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 13, "Using the Utility Panels to Encode and Decode Data."

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, which correspond to the functions they perform:

- DES master key
- AES master key
- PKA master keys
- Asymmetric master key on the PCICC, PCIXCC, CEX2C, or CEX3C.
- Data-encrypting keys
- Data-translation keys - not supported on the PCIXCC, CEX2C, or CEX3C
- MAC keys
- PIN keys
- Cryptographic Variable keys
- Transport keys
- Key Generating keys
- PKA keys

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.

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

ICSF uses master keys to protect keys that are used with the PCICC, PCIXCC, CEX2C, or CEX3C:

DES Master Key

The DES (DES-MK) master key is a double-length (128-bit) key that is used to protect symmetric DES/TDES keys used on the PCICC, PCIXCC, CEX2C or CEX3C. On a PCICC, this key must have the same value as the DES master key on the zSeries.

AES Master Key

The AES (AES-MK) master key is a 32-byte (256 bit) key that is used to protect keys used on the CEX2C or CEX3C. It is only available on the z9 EC, z9 BC, z10 EC, and z10 BC with the Nov. 2008 or later licensed internal code (LIC).

Asymmetric Master Key

The asymmetric (ASYM-MK) master key is a triple-length (192-bit) key. The ASYM-MK master key protects PKA private keys that are used on the PCICC, PCIXCC, CEX2C, or CEX3C.

Restriction: Master keys on a z990 or z890 require a PCIXCC or CEX2C. Master Keys on a z9 EC and z9 BC require a CEX2C. Master keys on a z10 EC and z10 BC require a CEX2C or CEX3C.

ICSF uses three types of master keys to protect keys that are used with the Cryptographic Coprocessor Feature:

DES Master Key

The DES master key is a double-length (128-bit) key that is used to protect DES and CDMF keys.

PKA Key Management Master Key

The PKA key management master key (KMMK) is a triple-length (192-bit) key. The KMMK protects PKA private keys that are used in both the digital signature services and in the CDMF and DES data key distribution functions. Support for the PKA KMMK is available only on the Cryptographic Coprocessor Feature on the IBM @server zSeries 900 processors.

PKA Signature Master Key

The PKA signature master key (SMK) is a triple-length (192-bit) key. The SMK protects PKA private keys that are used only in digital signature services. Support for the PKA SMK is available only on the Cryptographic Coprocessor Feature on the IBM @server zSeries 900 processors.

Note: On CCF systems, it is strongly recommended that the KMMK have the same value as the SMK.

Data-Encrypting Keys

Data-encrypting keys, also referred to as data keys, are used to encrypt and decrypt data. AES and DES data-encrypting keys are supported. DES keys can be

single-length, double-length, or triple-length. AES keys can be 128-bits, 192-bits, or 256-bits in length. Data keys can be either encrypted under the master key or in the clear.

Single-length DES data-encryption keys can also be used in place of the MAC keys to generate or verify a message authentication code.

CIPHER keys are DES data-encrypting keys (CIPHER, ENCIPHER, and DECIPHER). They are single- or double-length keys and can be used only for encrypting or decrypting data.

Data-Translation Keys

Data-translation keys are single-length (64-bit) keys that 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 data-translation key is reenciphered under another data-translation key on the intermediate node. During this process, the data never appears in the clear.

A data-translation key cannot be used in the decipher callable service to decipher data directly. It can translate the data from encipherment under one data-translation key to encipherment under another data-translation key. See “Protection of Data” on page 22 for a description of how data-translation keys protect data that is sent through intermediate systems.

Restriction: Data-translation keys are not supported on the PCIXCC, CEX2C, or CEX3C.

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 are either single-length (64-bit) or double-length (128-bit) keys.

A DES MAC key or an AES DATA key checks that a message you receive is the same one the message originator sent.

Note: For CCF/PCICC systems only. In order to generate and use double-length MAC keys in importable or exportable form, the CKDS must contain NOCV-enablement keys and ANSI system keys. When creating a new CKDS, add the NOCV-enablement keys and ANSI system keys during the initialization process. For information on initializing a CKDS, refer to “Initializing the CKDS and PKDS at First-Time Startup” on page 110.

ICSF uses these MAC keys in message authentication:

MAC Generation Keys

Before sending a message, an application program can generate an authentication code for the message, using the MAC generate callable service. The callable service computes the message authentication code by using a MAC generation key to process the message text. The originator of the message sends the message authentication code with the message text.

Single-length MAC generation keys (MAC keys) are used in the ANSI X9.9-1 MAC procedure. They support EMV algorithms. Double-length MAC generation

keys (DATAM keys) are used in the ANSI X9.19 optional double key MAC procedure. For compatibility with ICSF Version 2 Release 1, ICSF continues to support the MACD key type, which uses the single-length control vector for both the left and right half of the key to create an external token (MAC || MAC).

On the z990, z890, z9 EC, z9 BC, z10 EC, and z10 BC, ICSF supports double-length MAC keys with the MAC key type.

MAC Verification Key

The message receiver uses a single-length (MACVER) or double-length (DATAMV) MAC verification key to verify the message authentication code that the message originator sends.

Note: On the z990, z890, z9 EC, z9 BC, z10 EC, and z10 BC, ICSF supports double-length MACVER keys with the MACVER key type.

When the receiver gets the message, an application program calls the MAC verify callable service. The callable service verifies a message authentication code by using the MAC verification key to process the message text. It compares the MAC it generates internally with the MAC that was sent with the message. If the two MACs are the same, the message that was sent is identical to the message that was received.

The MAC generation key the sender uses and the MAC verification key the receiver uses have the same clear value. However, each is protected under the master key variant for its key type.

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. All PIN keys are double-length (128-bit) DES keys.

PIN keys for generating and verifying PINs and PIN offsets

These PIN keys generate and verify PINs and PIN offsets:

PIN Generation Key

A PIN generation key is used in an algorithm to generate PINs or PIN offsets.

To generate PINs, use an application program to call the PIN generate callable service. The PIN generation algorithm uses the PIN generation key and some relevant data to generate a clear PIN, a PIN verification value, or an offset.

PIN Verification Key

A PIN verification key is used in an algorithm to verify PINs and PIN offsets.

To verify a supplied PIN, use an application program to call the PIN verification callable service. You need to specify the supplied enciphered PIN block and PIN-encrypting key that enciphers it. You must also specify the PIN verification

key, the PIN verification algorithm, and other relevant data. The callable service generates a verification PIN. It compares the supplied PIN and the verification PIN, and if they are the same, it verifies the supplied PIN.

For a specific PIN generation key and PIN verification key pair, the PIN generation key and the PIN verification key have the same clear value. However, each key is protected by the master key variant for its key type.

PIN keys to protect and translate PIN blocks

These PIN keys protect and translate PIN blocks:

Output PIN-Encrypting Key

Two systems must share a common key for securely transmitting PIN blocks. The output PIN-encrypting key protects PIN blocks that are sent from your system to another system.

PIN-encrypting keys are used in the PIN translate service. Use the PIN translate service to translate PIN blocks from protection under one PIN-encrypting key to protection under another PIN-encrypting key. You can also use the PIN translate service to translate a PIN block from one PIN block format to another PIN block format. For more information about the PIN translate service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Input PIN-Encrypting Key

Two systems must share a common key for securely transmitting PIN blocks. The input PIN-encrypting key protects PIN blocks that are sent from another system to your system.

PIN-encrypting keys are used in the PIN translate service. You also use the input PIN-encrypting key in the PIN verify service. For more information about the PIN translate service and PIN verify service, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

For a specific pair of PIN-encrypting keys, the input PIN-encrypting key and the output PIN-encrypting key have the same clear value. However, each key is protected by the master key variant for its key type.

Cryptographic Variable Keys

These single or double-length DES keys are used to encrypt special control values in CCA DES key management. The Control Vector Translate and Cryptographic Variable Encipher callable services use cryptographic variable encrypting keys.

Transport Keys

Transport keys protect a key that is sent to another system, received from another system, or stored with data in a file. Transport keys are double-length (128-bit) DES keys.

These transport keys support the Common Cryptographic Architecture:

Exporter or OKEYXLAT Key-encrypting Key

An exporter or OKEYXLAT 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. Exporter key-encrypting keys are double-length keys. An exporter key is paired with an importer or IKEYXLAT key-encrypting key.

Importer or IKEYXLAT Key-encrypting Key

An importer or an IKEYXLAT 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. Importer key-encrypting keys are double-length keys. An importer key is paired with an exporter or OKEYXLAT key-encrypting key.

For a specific pair of transport keys, the importer key-encrypting key and the exporter key-encrypting key have the same clear value. However, each key is protected by the master key variant for its key type.

ICSF provides this transport key type to support the ANSI X9.17 standard.

ANSI Key-encrypting Key

An importer and exporter key-encrypting key that is used in the ANSI key management callable services. ANSI key-encrypting keys (AKEKs) are bidirectional and are either single- or double-length keys.

Restriction: ANSI keys are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900.

Key Generating Keys

Key-generating keys are double-length keys used to derive unique-key-per-transaction DES keys.

PKA Keys

ICSF supports the use of public key cryptography. This requires the generation of a pair of PKA keys. One key is made public, and the other key is kept private. The private key is protected through encryption under the appropriate PKA master key. The public key is used to encrypt DES or AES data-encrypting keys in a key distribution system. The private key is then used to decrypt the DES or AES data-encrypting key. The private key is also used for generating digital signatures which are verified using the corresponding public key.

ICSF supports the use of these PKA keys.

RSA

An RSA key pair includes a private key and a public key. RSA keys can be used for key distribution and authentication. When used for key distribution, a DES key is encrypted under an RSA public key by the sender. The key can only be decrypted with the receiver's private key. When used for authentication, the RSA private key is used for digital signature generation and the RSA public key is used for digital signature verification.

The optional PCICC, PCIXCC, CEX2C, or CEX3C provide the ability to generate RSA public and private key pairs within their secure hardware boundary.

The Cryptographic Coprocessor Feature (CCF) does not provide the ability to generate RSA public and private keys within its secure hardware boundary. If you have CCF without a PCI Cryptographic Coprocessor, you can generate RSA key pairs in the encrypted form on a TKE Workstation with APAR OW32982 or a workstation with a 4764 or 4758 cryptographic adapter installed. RSA keys generated on the TKE workstation can be loaded directly to the PKDS from the TKE workstation. RSA keys generated on a non-TKE

workstation can use the PKA key import callable service to import the RSA key pair to the Cryptographic Coprocessor Feature.

DSS

A DSS key pair also includes a private and a public key. The DSS private key is used for digital signature generation, and the DSS public key is used for digital signature verification.

ICSF provides a callable service to generate PKA internal key tokens for use with the DSS algorithm in digital signature services.

Restriction: DSS keys are not supported on the PCIXCC, CEX2C, or CEX3C.

RSA and DSS public and private keys can be stored in the PKA key data set (PKDS), a VSAM data set. For retained private keys, only the public key is stored in the PKDS. For more information about the PKDS, refer to “Setting up and maintaining the PKDS” on page 36.

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 and AES data-encrypting keys that are transported between systems.

ICSF controls the use of DES keys by separating them into types that can be used to do only specific functions. AES keys are not separated into types.

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.

Key Separation

The cryptographic hardware, or cryptographic feature, controls the use of 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.

Depending on the cryptographic feature, an ICSF system may have multiple master keys:

- A DES master key protecting keys that are used in DES or CDMF operations on the Cryptographic Coprocessor Feature.
- A DES master key protecting keys that are used in operations on the PCICC, PCIXCC, CEX2C, or CEX3C
- An AES master key protecting keys that are used in operations on the CEX2C or CEX3C.
- A PKA key management master key (KMMK) protecting keys that are used in PKA key distribution operations on the Cryptographic Coprocessor Feature.
- A PKA signature master key (SMK) protecting keys that are used in digital signature operations on the Cryptographic Coprocessor Feature.
- An asymmetric-keys (ASYM-MK) master key protecting RSA keys used in key distribution and authentication operations on the PCICC, PCIXCC, CEX2C, or CEX3C.

DES master key variants protect DES and CDMF keys

To provide for key separation, the cryptographic feature automatically encrypts each type of key that is used in either DES or CDMF services 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 or CDMF services.

Restriction: CDMF services are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900.

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 2 on page 17, 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 feature 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 feature 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 or CDMF services has a unique control vector associated with it. For example, the cryptographic feature 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.

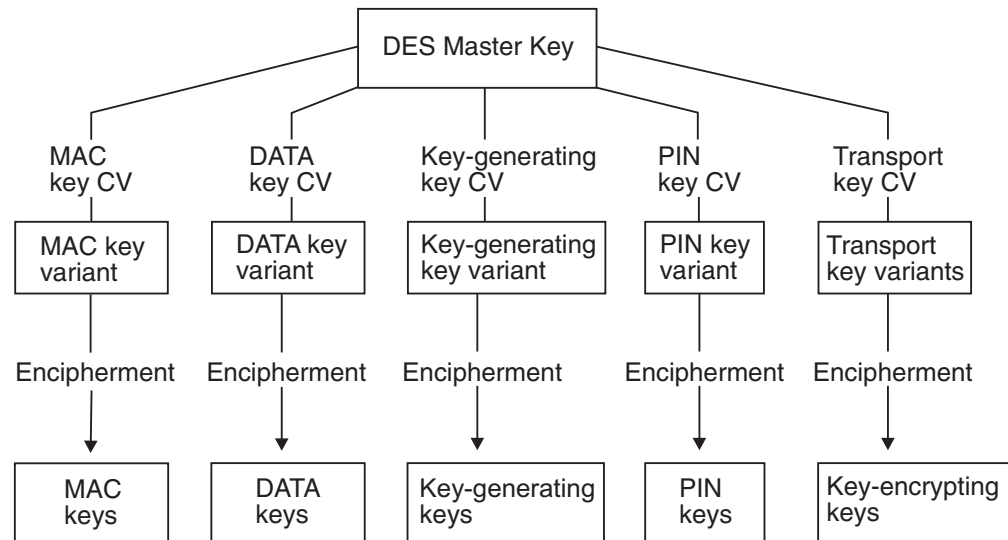


Figure 2. Keys Protected in a System

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" for a listing of the control vector that is used for each key type.

DES cryptographic keys can be single- 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. 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.

Multiple Encipherment

The cryptographic feature uses multiple encipherment when it enciphers a key under a key-encrypting key such as the master key or a transport key. Multiple encipherment is used whenever the key-encrypting key is double-length. The cryptographic feature enciphers each half of the key that it is encrypting.

To multiple-encipher the left half of a key, the cryptographic feature performs these steps:

1. Exclusive ORs the left half of the key-encrypting key with the control vector for the left half of the key to create the variant. The cryptographic feature then enciphers the left half of the key under this variant.

2. Exclusive ORs the right half of the key-encrypting key with the control vector for the left half of the key to create the variant. The cryptographic feature then deciphers the value that results from step 1 on page 17 under this variant.
3. Exclusive ORs the left half of the key-encrypting key with the control vector for the left half of the key. The cryptographic feature then enciphers the value that results from step 2 under this variant.

To multiple-encipher the right half of the key, the cryptographic feature performs these steps:

1. Exclusive ORs the left half of the key-encrypting key with the control vector for the right half of the key to create the variant. The cryptographic feature then enciphers the right half of the key under this variant.
2. Exclusive ORs the right half of the key-encrypting key with the control vector for the right half of the key to create the variant. The cryptographic feature then deciphers the value that results from step 1 under this variant.
3. Exclusive ORs the left half of the key-encrypting key with the control vector for the right half of the key. The cryptographic feature then enciphers the value that results from step 2 under this variant.

On ICSF, an effective single-length key can exist as a double-length key; each key half has an identical value. The result of the multiple encipherment process on an effective single-length key is the key value that is encrypted once under the variant.

Migrating from PCF Key Types

Your installation may use Programmed Cryptographic Facility (PCF). ICSF provides key types that are similar to the PCF key types and provides other key types for enhanced key separation and more functions. You cannot use a PCF key on ICSF, but you can convert a PCF key into an ICSF key. Table 1 lists which ICSF key types correspond to the PCF key types.

Table 1. PCF and Corresponding ICSF Key Types

PCF 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 with ICSF and to migrate from PCF to ICSF. The conversion program converts PCF keys to ICSF keys. For information about migration from PCF to z/OS ICSF, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

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 enciphered under a DES transport key.
- DES and AES keys enciphered under the receiver's RSA public key.

When ICSF enciphers a key under a DES 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 Chapter 4, "Using RACF to Protect Keys and Services," on page 43.

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 3, you use an importer key-encrypting key to encrypt the data-encrypting key.

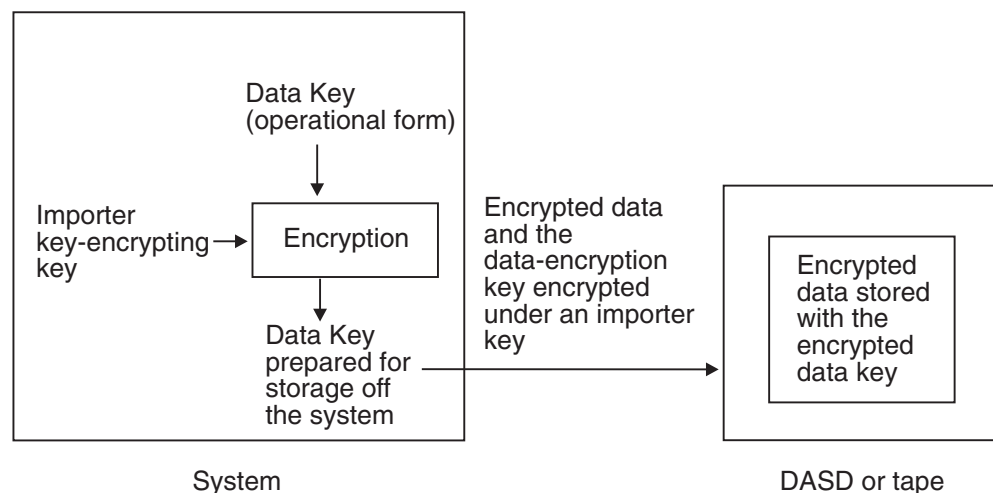


Figure 3. Keys Protected in a File Outside the System

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

Using DES 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 4 on page 21, when you send the PIN between systems, you encipher the PIN under a PIN-encrypting key.

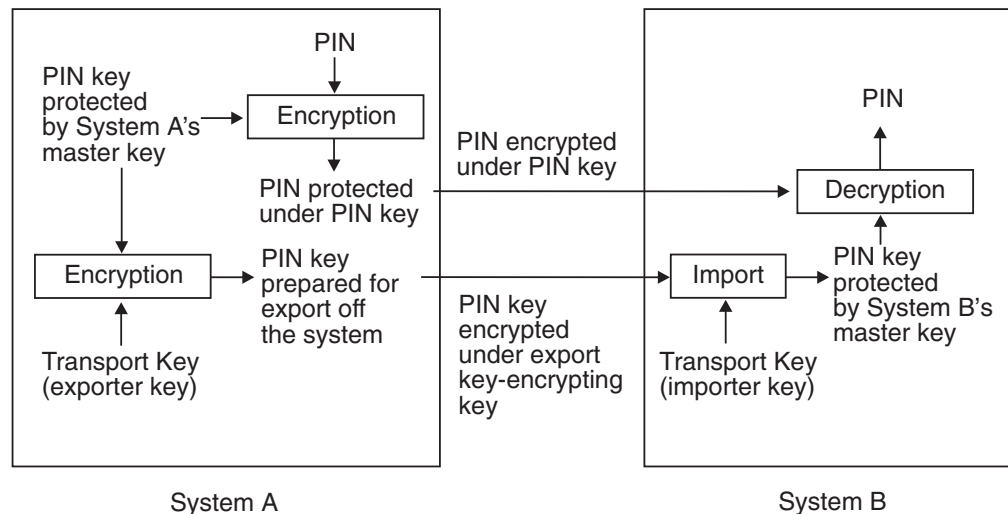


Figure 4. Keys and PINs Protected When Sent between Two Systems

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 DES or AES cryptography, it is not the method of choice for all cryptographic functions. It can be used, however, in combination with DES and AES cryptography to enhance the security of key exchange. DES data-encrypting keys and AES data-encrypting 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 data-encrypting keys. An example of this is shown in Figure 5. The sending system enciphers the data-encrypting key under the receiver's RSA public key and sends the enciphered data-encrypting key to the receiver. The receiver uses his or her RSA private key to decipher the data-encrypting key.

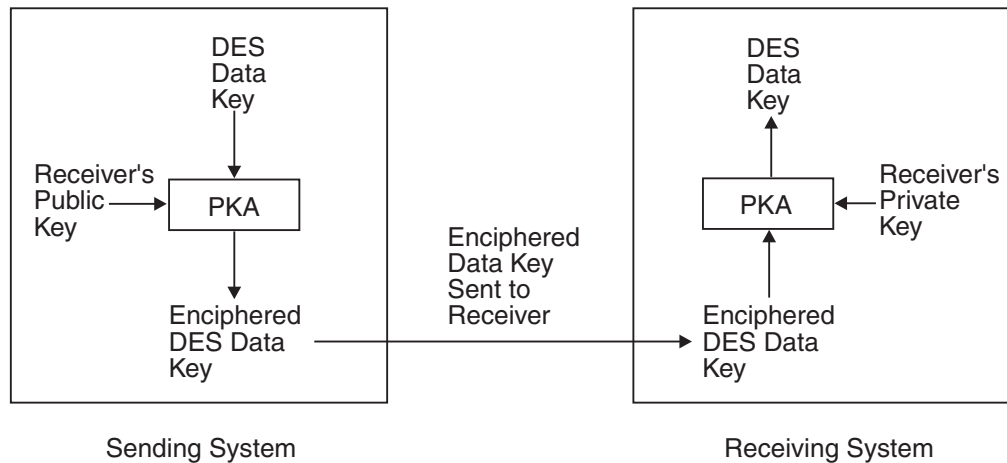


Figure 5. Distributing a DES Data-Encrypting Key Using an RSA Cryptographic Scheme

Note: Only data-encrypting keys can be encrypted under RSA public keys.

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 6 on page 23.

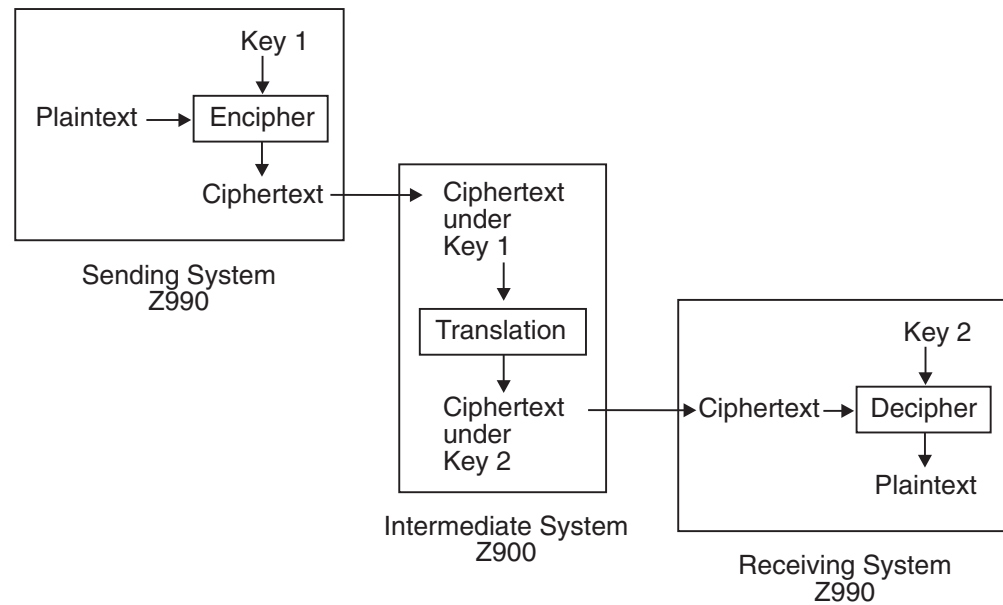


Figure 6. Data Protected When Sent between Intermediate Systems

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 data-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 data-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 6, 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 6, this key is called *Key 2*. Note that *Key 1* and *Key 2* do not have the same clear key value.

The data-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.

Data-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/data-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 data-translation/data-encrypting key pair.

Triple DES for Privacy

ICSF supports triple DES encryption for data privacy. This provides stronger encryption than the current DES algorithm and single-length DES data-encryption keys. Triple DES uses three, single-length keys to encipher and decipher the data which results in a stronger form of cryptography.

Data that has been encrypted under a double-length or triple-length DATA key cannot be reenciphered using data-translation keys as described in “Protection of Data” on page 22.

Advanced Encryption Standard (AES)

ICSF supports the Advanced Encryption Standard (AES) algorithm for data privacy. This provides strong encryption. Data can be encrypted and decrypted using 128-bit, 192-bit, and 256-bit keys. The algorithm has the same availability as triple DES.

AES on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC requires feature 3863, CP Assist for Cryptographic Functions (CPACF) DES/TDES Enablement.

Chapter 3. Managing Cryptographic Keys

To perform cryptographic services, you need to know how to create, maintain, and use cryptographic keys. This topic gives an overview on entering master keys, generating keys, creating and maintaining the cryptographic key data sets (CKDS, PKDS and TKDS), and entering keys into the CKDS. This topic also discusses distributing keys and controlling access to keys.

Generating Cryptographic Keys

Using ICSF, you can generate keys by using either the key generator utility program (KGUP) or the key generate callable service. Both KGUP and the key generate callable service create all types of keys except PKA keys and ANSI X9.17 keys. 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 dynamic CKDS update service to store the key in the CKDS.

Enhanced key management for crypto assist instructions

Certain ICSF callable services exploit DES and AES instructions on the CPACF for performance. These services accept a clear key value or a clear key token. The clear key tokens can be stored in the CKDS. These services have been enhanced to accept both encrypted AES and DES key tokens. These encrypted tokens must be stored in the CKDS.

With clear key support on the CKDS, clear keys do not have to appear in application storage during use. Clear key tokens in the CKDS can be referenced by label name in these callable services:

- Symmetric Key Encipher (CSNBSYE, CSNBSYE1, CSNESYE and CSNESYE1)
- Symmetric Key Decipher (CSNBSYD, CSNBSYD1, CSNESYD and CSNESYD1)
- Symmetric MAC Generate (CSNBSMG, CSNBSMG1, CSNESMG, and CSNESMG1)
- Symmetric MAC Verify (CSNBSMV, CSNBSMV1, CSNESMV, and CSNESMV1)

The following callable services have been enhanced to support encrypted data-encrypting keys from the CKDS:

- Symmetric Key Encipher (CSNBSYE, CSNBSYE1, CSNESYE and CSNESYE1)
- Symmetric Key Decipher (CSNBSYD, CSNBSYD1, CSNESYD and CSNESYD1)

The configuration needed to enable an encrypted key in the CKDS to be used in these callable services involves modifying the profile in the CSFKEYS class that covers the key to indicate that ICSF can rewrap the encrypted key using the CPACF wrapping key. In order to use an encrypted key in these services, this rewrapping is necessary. However, although the rewrapping operation ensures that the key is not visible in application or system storage, it does move the key outside of the tamper-resistant hardware boundary of the cryptographic card which may violate your installation's security policy. For more information, refer to "Enabling use of encrypted keys in Symmetric Key Encipher and Symmetric Key Decipher callable services" on page 71.

On systems sharing the CKDS without this support, it is highly recommended that you RACF-protect the label name of the clear key tokens on the other systems. This will provide additional security for your installation.

TKDS key protection

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

Generating PKA Keys

If a PCICC, PCIXCC, CEX2C, or CEX3C is installed, ICSF is able to generate RSA keys using the PKA Key Generate service. The RSA key format can be the Modulus Exponent form or the Chinese Remainder form. Retained keys are RSA keys 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. Retained key format can be the Modulus Exponent form or the Chinese Remainder form. 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 recreate 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 Permit Regeneration Data for Retain Keys access control point.
- When not using the RETAIN keyword, enable the Permit Regeneration Data access control point.

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

PKA keys in the PKDS can be managed using the PKA 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 14, "Using the Utility Panels to Manage Keys in the PKDS," on page 313.

Key Generator Utility Program (KGUP)

You can use KGUP to generate keys in either an operational form or an exportable form. When KGUP generates a key in the operational form, it stores it in the cryptographic key data set (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 10, "Managing Cryptographic Keys by Using the Key Generator Utility Program," on page 191.

Key Generate Callable Service

The key generate callable service generates a single key or a pair of 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.

When you call the key generate callable service, you pass parameters that specify information about the key you want generated. The key generate callable service 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

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

For more information about the key generate callable service, see *z/OS Cryptographic Services ICSF Application Programmer'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.

- A DES master key on the Cryptographic Coprocessor Feature protects DES keys and PKA master keys protect DSS and RSA keys.
- On the PCICC, PCIXCC, CEX2C, or CEX3C, the DES master key (DES-MK) protects DES keys and the asymmetric-keys master key (ASYM-MK) protects RSA keys.
- On the CEX2C or CEX3C, the AES master key (AES-MK) protects AES keys.

The first time you start ICSF on your system, you may enter master keys and initialize the cryptographic key data set (CKDS) and PKA cryptographic key data set (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 the PKDS.

If you have no coprocessor, you can initialize the CKDS for use with clear AES and DES data keys. This CKDS can not 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 feature. 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 feature.

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 the DES or AES master key periodically

- Changing the PKA 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.

You can use any of these methods to enter the master keys:

- Pass Phrase Initialization

The pass phrase initialization utility allows the user of ICSF to:

- set both the DES and PKA master keys on the Cryptographic Coprocessor Feature, PCICC and PCIXCC.
- set the DES-MK, AES-MK, and ASYM-MK on the CEX2C or CEX3C
- initialize the CKDS and PKDS

For steps in using the pass phrase initialization utility, refer to Chapter 5, “Using the Pass Phrase Initialization Utility,” on page 73.

- 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 cryptographic coprocessor hardware. 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 and complexity of the optional TKE workstation. For master key entry steps on the coprocessors, see Chapter 6, “Managing Master Keys - CCF and PCICC,” on page 93 and Chapter 7, “Managing Master Keys - PCIXCC, CEX2C, or CEX3C,” on page 135.

- 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 Cryptographic Coprocessor Features and Cryptographic Coprocessors within a server complex.

You must use TKE V4.0 or higher to set up DES master keys on a PCIXCC/CEX2C. You must use TKE V5.3 or higher to set up AES master keys on a CEX2C. You must use TKE V6.0 or higher to set up AES master keys on a CEX3C.

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

When you have entered the master keys, choose option 1 on the ICSF Initialize a CKDS panel to:

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

If you wish to add ANSI, NOCV, or Enhanced System Keys to your CKDS, choose the appropriate option. Refresh the CKDS. Note that these keys are not present in

a CKDS initialized on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC. A CKDS initialized on the newer systems (listed above) cannot be shared with legacy systems.

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

When you have entered the PKA master keys, enter the name of the PKDS to be initialized on the panel. Choose option 5, INITIALIZE PKDS, on the ICSF Master Key Management panel.

Entering system keys into the cryptographic key data set (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 CKDS. Other keys are optional; however, their absence will affect functions in many services.

Note: The NOCV, ANSI and Extended Systems keys are not required on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system with a PCIXCC, CEX2C, or CEX3C.

If the system keys are not in the CKDS, an 18F abnormal end with reason code X'A1' can occur. If the ANSI, NOCV enablement, or the ESYS keys are not in the CKDS, an 18F abnormal end with reason code X'A3' can occur.

This is a summarization of where the keys are used:

- Required System Keys

These keys are used to validate CKDS entries and used in many services. These keys are required.

- NOCV-enablement Keys

- These keys are needed for all services where NOCV key-encrypting keys are required. See *z/OS Cryptographic Services ICSF Application Programmer's Guide* for more information.
- These keys are needed in CSNBKGN and KGUP where replicated keys are generated, that is, where key length of SINGLE is specified for double-length keys.
- These keys are used during verification pattern generation on a CDMF-only system.
- These keys are used by CSNBSBC on a CDMF-only system.
- These keys are used during CKDS conversion.
- These keys are required to export and import double-length DATAM and DATAMV keys.

- ANSI System Keys

- These keys are used by CSNBSBD on a CDMF-only system.
- These keys are used when installing the extended system keys (ESYS) on the CKDS initialization panel.
- These keys are needed for key part import services.
- These keys are required for key test service CSNBKYT if there are no PCICCs active.
- These keys are required to generate double-length DATAM and DATAMV keys in the importable form.

- Extended System Keys

These keys are required for symmetric key export if there are no PCICCs active.

Entering keys into the cryptographic key data set (CKDS)

All DES and AES keys except the 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)

Regardless of your processor or server model, you can use KGUP to enter keys into the CKDS.

- Dynamic CKDS update callable services

Regardless of your processor or server model, you can program applications to use the dynamic CKDS update callable services to enter keys into the CKDS.

- Trusted Key Entry (TKE) workstation

With the TKE workstation you can load key parts for operational (PIN and transport) keys into a key queue on the CCF. To load these key parts into the CKDS, you must also use the ICSF Operational Key panel and perform a CKDS refresh. For more information, refer to *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

DES operational key support for PCIXCC/CEX2C is available in TKE V4.1 and higher. AES operational key support for CEX2C is available in TKE V5.3. 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 DES Operational Key Load panel or KGUP. For more information, refer to the *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

The table in Table 2 shows which keys can be entered by each of these methods.

Table 2. Methods for Entering Each Key Type into the CKDS

Key Type	KGUP	Dynamic Update	TKE with CCF	TKE with PCIXCC, CEX2C, or CEX3C
PIN	X	X	X	X
Importer and Exporter key-encrypting keys	X	X	X	X
Data-encrypting	X	X		
Data-translation*	X	X		
MAC and MACVER	X	X		X
DATAM and DATAMV	X	X		X
ANSI key-encrypting keys*		X		
IMP-PKA keys	X**	X	X	X
Non-standard CV keys	X**	X		X

Notes:

1. * ANSI and data-translation keys are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900.
2. ** The key can only be loaded thru the KGUP OPKYLOAD option, requiring a TKE workstation to accumulate the key in the key part register.

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, several conditions must be met.

- The installation options data set must specify YES for the SSM installation option.

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

- The environmental control mask (ECM) must be configured to permit special secure mode.

The ECM is a 32-bit mask that is defined for each crypto domain during hardware installation. The second bit in this mask must have been turned on to enable special secure mode.

This is required for systems with the Cryptographic Coprocessor Feature.

- If you are running in LPAR mode, special secure mode must be enabled

You enable special secure mode during activation using the Crypto page of the Customize Activation Profiles task. After activation, you can enable or disable special secure mode on the Change LPAR Crypto task. Both of these tasks can be accessed from the Hardware Master Console.

This is required for systems with the Cryptographic Coprocessor Feature.

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 10, "Managing Cryptographic Keys by Using the Key Generator Utility Program," on page 191.

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 these services to create, write, and delete records 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 callable service 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 service, refer to *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Entering keys into the PKDS

You can store RSA and DSS public and private keys in the PKA key data set (PKDS). Trusted block tokens can also be stored in the PKA key dataset through the use of the same ICSF callable services. ICSF provides a set of callable services that allow applications to update the PKDS. Applications can use some of these services to create, write, and delete records from the PKDS. ICSF maintains an in-storage copy of the PKDS similar to the in-storage copy of the CKDS. It's purpose is to improve performance and eliminate I/O.

Restriction: DSS keys are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900.

For more information on using the PKDS update services, refer to the *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

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

Besides the in-storage PKDS, there is a copy of the PKDS on disk. Your installation can have many disk copies of PKDSs, 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. However, only one disk copy is read into storage at a time.

RSA private keys can also be stored in the PKDS from TKE. For more information, see *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

Entering cryptographic objects into the TKDS

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 a specialized data set called the token data set (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 public key objects, private key objects, secret key objects, certificate objects, data objects, and domain parameter objects in the TKDS through the use of ICSF callable services. ICSF provides a set of callable services that allow applications to update the TKDS. Applications can use these services to create, delete, list, set and get attribute values from the TKDS.

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

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 Security Requirements for

Cryptographic Modules standard (FIPS 140-2), 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 TKDS services, refer to the *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* and the *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Maintaining cryptographic keys

You can use either KGUP or the dynamic CKDS update services to generate and enter keys into the cryptographic key data set (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 10, “Managing Cryptographic Keys by Using the Key Generator Utility Program,” on page 191.

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 RACF to control which applications can use specific keys and services. For more information, see Chapter 4, “Using RACF to Protect Keys and Services,” on page 43.

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

The cryptographic key data set (CKDS) stores operational DES and AES keys of all types. It contains an entry for each key.

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

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. Encrypted keys in the CKDS can not be overwritten with a key encrypted under a different master key. (DES replaces DES, AES replaces AES). For clear keys, the same is true, DES can overwrite DES, AES can overwrite AES.

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 6, “Managing Master Keys - CCF and PCICC,” on page 93 or Chapter 7, “Managing Master Keys - PCIXCC, CEX2C, or CEX3C,” on page 135.

Once 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 10, “Managing Cryptographic Keys by Using the Key Generator Utility Program,” on page 191. 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 use the optional TKE workstation to load key parts for operational (PIN and transport) keys into a key part queue on the CCF. To load these key parts into the CKDS, you must also use the ICSF Operational Key panel and perform a CKDS refresh. For more information on using the TKE workstation, see *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

Support for operational keys is available beginning with TKE V4.1 for the z990, z890, z9 EC, z9 BC, z10 EC and z10 BC. You can load key parts for all operational keys into key part registers on the PCIXCC, CEX2C, or CEX3C. To load the accumulated key into the CKDS, you must use the ICSF Operational Key Load panel. For more information, refer to the *z/OS Cryptographic Services ICSF TKE PCIX 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 disk copies of CKDSs, backup copies, and different disk copies. For example, an installation may 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.

A CKDS with encrypted AES keys must be managed from a system that has an AES master key.

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 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 7 on page 35 shows that ICSF callable services use keys in the in-storage copy of the CKDS.

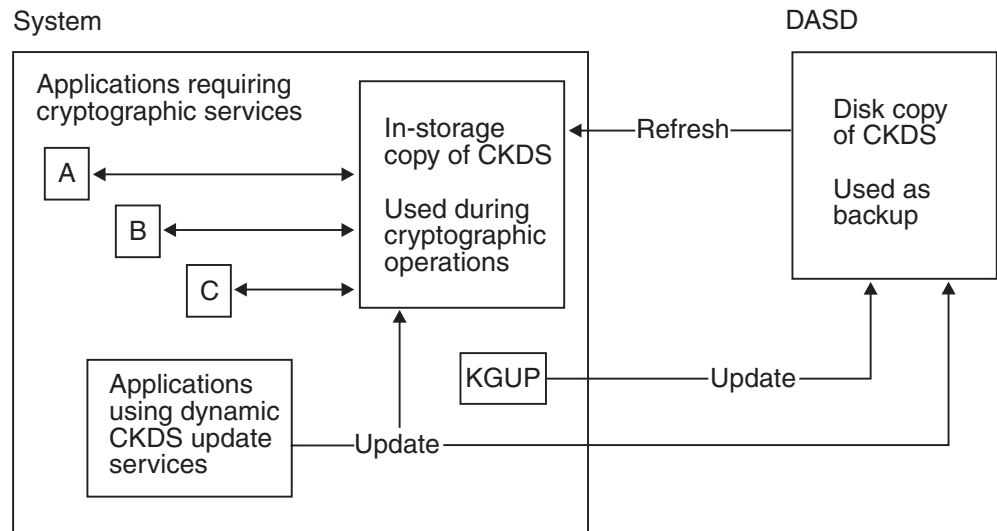


Figure 7. Updating the In-Storage Copy and the Disk Copy of the CKDS

You just 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 master key. To change the 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 key. Therefore, to make the new master key active, the CKDS must be reenciphered from under the current master key to under the new master key.

First, you reencipher the disk copy of the CKDS under the new master key. Then you activate the new master key 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 key. If you have multiple disk copies of CKDSs, reencipher all of them under the new master key before changing the master key.

You can reencipher a CKDS under a new master key by using the master key panels or a utility program. For more information about reenciphering a CKDS, see “Steps for changing the DES master key and reenciphering the CKDS” on page 121.

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, refer to “Steps for disallowing dynamic CKDS updates during KGUP updates” on page 192.

If running in a sysplex, see Chapter 9, “Running in a Sysplex Environment,” on page 181.

Setting up and maintaining the PKDS

RSA and DSS public and private keys, and trusted blocks can be stored in the PKA key data set (PKDS), a VSAM data set. Applications can use the dynamic PKDS callable services to create, write, read and delete PKDS records.

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 PKA master keys. Internal tokens, however, can only be used if they are encrypted under the current PKA signature master key (SMK), the key management master key (KMMK), or the asymmetric-keys master key (ASYM-MK).

Your installation should periodically change the PKA/asymmetric master key. To change the master key, you enter a new master key value and make that value active. After the master key has been set, the PKDS must be reenciphered under the new master key. You can reencipher a PKDS under a new master key by using the master key panels or a utility program. For more information about reenciphering a PKDS, see “Steps for reenciphering and refreshing the PKDS” on page 129. If you have multiple disk copies of PKDSs, reencipher all of them under the new master key after changing the master key.

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 9, “Running in a Sysplex Environment,” on page 181.

Distributing Cryptographic Keys

With ICSF you can develop key distribution systems as defined in any of these:

- The IBM Common Cryptographic Architecture
- The ANSI X9.17 Standard
- The Public Key Cryptographic Standard (PKCS)

These key distribution systems are explained in the these topics:

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.

Complementary keys apply only to DES keys. AES keys can be distributed to other systems using RSA key pairs.

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 data-translation key (**Restriction:** Data-translation keys are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900).
- 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 data-translation key (**Restriction:** Data-translation keys are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900).
- Data-translation key and data-translation key (**Restriction:** Data-translation keys are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900).

For more information about this situation, see “Protection of Data” on page 22.

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 8 on page 38 System A wants to send an output PIN-encrypting key to System B.

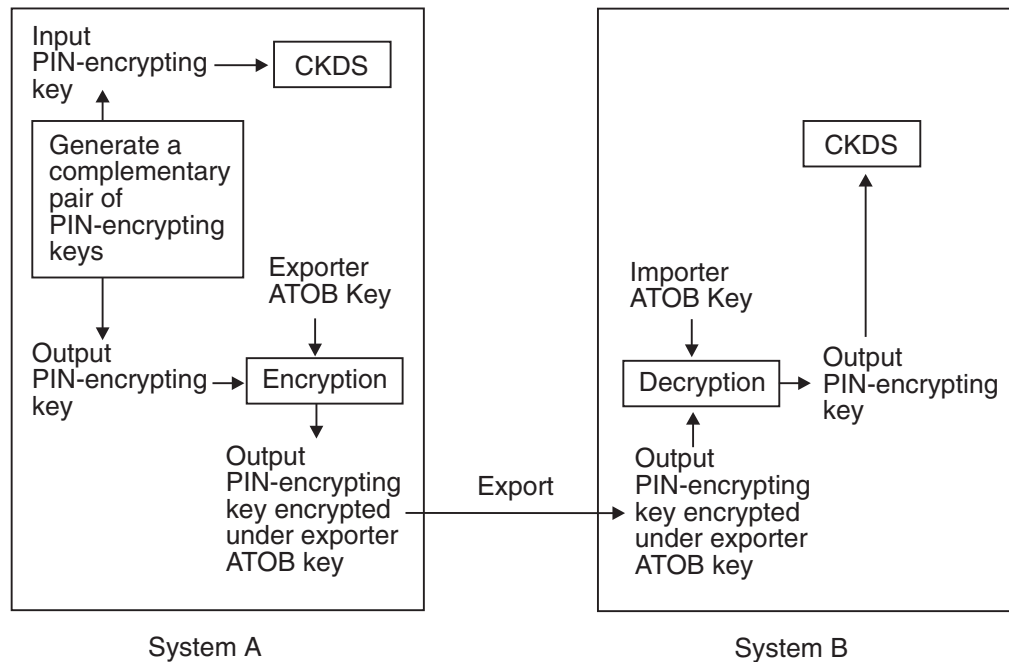


Figure 8. Key Sent from System A to System B

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 9 on page 39.

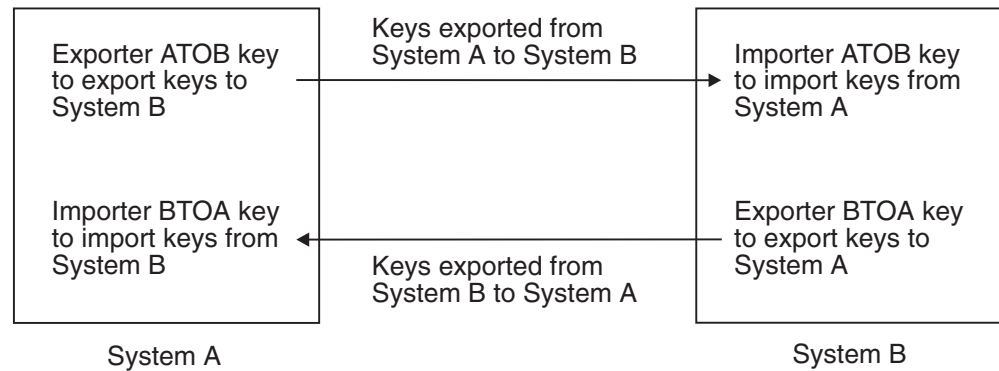


Figure 9. Keys Sent between System A and System B

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 10, “Managing Cryptographic Keys by Using the Key Generator Utility Program,” on page 191.

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 22.

ANSI X9.17 Key Distribution

ICSF provides callable services that allow you to develop key distribution systems that adhere to the ANSI X9.17 standard.

Restriction: These services are not supported on a PCIXCC/CEX2C.

When protected data is sent between two systems, it is protected by data-encrypting keys. The same data-encrypting key exists on two different systems so that both systems can encipher and decipher the data.

For two systems to exchange keys, they must establish a shared transport key, the ANSI key-encrypting key (AKEK), which is distributed manually. This transport key is bidirectional, and can be used for distributing keys in both directions between System A and System B, as shown in Figure 10.

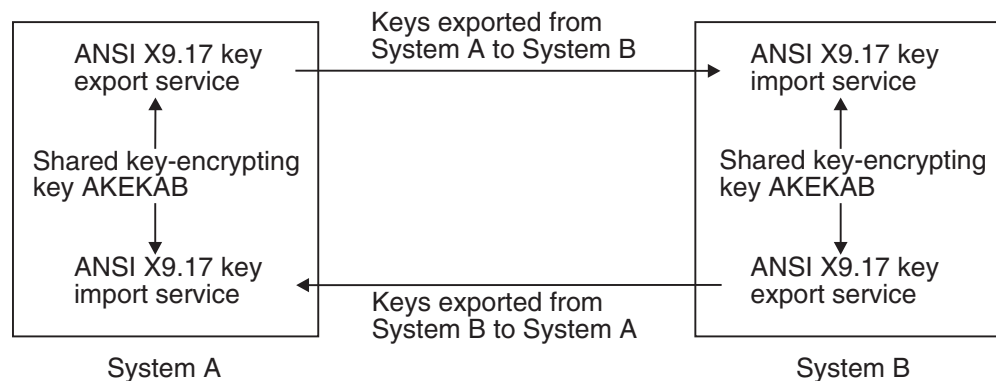


Figure 10. ANSI X9.17 Keys Sent between System A and System B

System A generates the data-encrypting key, enciphers it under System A’s master key, and stores it in the CKDS. System A uses the ANSI X9.17 key export callable service to encrypt the data-encrypting key under the shared transport key, AKEKAB, and export it to System B. System B then uses the ANSI X9.17 key import callable service to decrypt the data-encrypting key using the shared transport key, AKEKAB, and then encrypts it under System B’s master key. The shared transport key is coupled with source and destination identifiers for System A and System B, and a message counter as defined in the ANSI offset and notarization processes.

The shared ANSI key-encrypting key is bidirectional. System B can also send keys to System A. The systems can also exchange data keys along with the AKEK used to encrypt them. The AKEKs are themselves encrypted under the transport AKEK.

ANSI X9.17 key distribution can take place in several types of environments:

- Point-to-point environment
- Key distribution center environment
- Key translation center environment

For more information on ANSI X9.17 key distribution, refer to the ANSI X9.17 Standard.

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, AES and CDMF data-encrypting keys is presented in “Using RSA Public Keys to Protect Keys Sent between Systems” on page 21.

Controlling PCICC, PCIXCC, CEX2C, and CEX3C services

This topic only applies if you have a TKE workstation. For non-TKE users, all access control points are enabled with the exception of DKYGENKY-DALL and DSG ZERO-PAD unrestricted hash length and PTR enhanced PIN security. These are disabled for all users and require a TKE workstation to enable.

If you use TKE to administer your systems, new access control points must be enabled before the services are available.

Whether the various services are enabled or disabled on your system is dependent upon TKE workstation installation. Prior to TKE Version 3.1, only ISPF services could be updated. With TKE Version 3.1 and higher, access control points for API and UDX services on the PCI Cryptographic Coprocessor can be enabled/disabled.

To enable/disable access control points on a PCIXCC/CEX2C, TKE V4.0 or higher is required. To enable/disable access control points on a CEX3C, TKE V6.0 or higher is required.

If you have never installed a TKE workstation on your system, all access control points for ISPF and API (except those mentioned previously) will be enabled. (Note that for UDXs with access control points, enablement of UDX access control points requires a TKE workstation.)

To list the access control points that are enabled, see “Displaying PCICC default roles” on page 287 and “Displaying PCIXCC, CEX2C, and CEX3C default roles” on page 289.

If, however, you have previously installed a TKE Version 3 or higher workstation, your ISPF service settings will be the same as those for your existing system. The API settings will also be the same as your existing system, except for the new access control points (which are disabled). The UDX access control points would all be disabled.

As new access control points are added, they are enabled for new (first-time) TKE installations. For existing TKE installations, API services would reflect what had been enabled/disabled in Version 3.1 or higher, and new access control points would be disabled. 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 PCIX Workstation User's Guide*.

Chapter 4. Using RACF to Protect Keys and Services

You can use 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 RACF to audit the use of keys and services. In addition, you can establish a Key Store Policy that defines rules for the use of encrypted key tokens that are stored in a CKDS or PKDS. To use RACF to control access to keys and services, you create and maintain general resource profiles in the CSFKEYS class, the CSFSERV class, and the XFACILIT class.

- The CSFKEYS class controls access to 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. For the exclusive purpose of requiring UPDATE instead of READ authority when transferring a secure AES or DES 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 or CSNFSYX) is called. For all other callable services, the CSFKEYS class is used.
- The CSFSERV class controls access to ICSF services and ICSF TSO panel utilities.
- 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 a CKDS or PKDS can be accessed and used.

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

Steps for RACF-protecting keys and services

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

1. Decide whether you will protect keys, services, or both. You can select which keys and services to protect.
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 PIN-related callable services
- Users who work with a particular MAC, or a particular PIN
- Users who call applications to dynamically update the CKDS
- Users who perform functions available on the User Control Functions panel

Usually, all users of ICSF should have access to keys and services by virtue of their membership in one of these RACF groups, rather than specific users. This is because RACF 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 RACF data base is not sufficient. This

is analogous to the in-storage CKDS maintained by ICSF.) To refresh the in-storage RACF profiles, the RACF security administrator must use the SETROPTS command:

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

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

If you place *RACF groups* in the access lists of the RACF 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 RACF security administrator to create the RACF groups.

You should also ask your RACF 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 RACF 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 RACF 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 45 and “Setting up profiles in the CSFSERV general resource class” on page 46.
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 RDEFINE, RALTER, and SETR, see the *z/OS Security Server RACF Command Language Reference*.

7. Determine if you need to establish a Key Store Policy for a CKDS and/or a PKDS. 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. No key store policy controls are available or needed for a TKDS, because none of the keys in a TKDS are enciphered under a key-encrypting key. By enabling Key Store Policy controls, you can:
- verify, when an application passes a callable service a key token instead of a key label, that the user has authority to the secure token. Profiles in the CSFKEYS class are named based on key labels (a discrete profile will exactly match the key label, while a generic profile will contain generic characters to match a number of key labels). Because the profiles are based on the key label, a SAF authorization check needs to know the key label of a CKDS or PKDS key record in order to perform the authorization check. A Key Store Policy control is available that will, if an application passes a callable service a key token instead of a key label, locate the associated key label(s) for the token so that a SAF authorization check can be carried out. By default, if ICSF cannot find an associated key label for the key token, the callable service will fail. However, another Key Store Policy control lets you use a default profile to specify access authority to tokens that are not stored in the CKDS or PKDS.
 - prevent applications from storing duplicate tokens in a CKDS or PKDS.
 - raise the level of access authority required to create, write to, or delete a key label.
 - raise the level of access authority required to export a token using the Symmetric Key Export callable service (CSNDSYX or CSNFSYX).
 - set additional restrictions on how keys covered by the profile can be used.
- You should familiarize yourself with the controls you can enable and decide on the Key Store Policy that is best for your installation. Refer to “Defining a key store policy” on page 50 for more information.

Setting up profiles in the CSFKEYS general resource class

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

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 (this is not the transport key label). Note that if an application uses a token instead of a key label, no authorization checking is done on the use of the key.

Notes:

- a. If you have ICSF/MVS Version 1 Release 1 profiles that specify *key-type.label*, you need to change them to specify only *label*.
- b. As with any RACF 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.
- c. If you have already started ICSF, you need to refresh the in-storage profiles. See Step 3.
- d. You can specify other operands, such as auditing (AUDIT operand), on the RDEFINE or RALTER commands.

- e. If the RACF 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 RACF general resource class.
2. Give appropriate users (preferably groups) access to the profiles:


```
PERMIT profile-name CLASS(CSFKEYS)
      ID(groupid) ACCESS(READ)
```
3. When the profiles are ready to be used, ask the RACF security administrator to activate the CSFKEYS class and refresh the in-storage RACF profiles:


```
SETROPTS CLASSACT(CSFKEYS)

SETROPTS RACLIST(CSFKEYS) REFRESH
```

Setting up profiles in the CSFSERV general resource class

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

1. Define appropriate profiles in the CSFSERV class:

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

Where *service-name* is one of these:

CSFAEGN	ANSI X9.17 EDC generate callable service
CSFAKEX	ANSI X9.17 key export callable service
CSFAKIM	ANSI X9.17 key import callable service
CSFAKTR	ANSI X9.17 key translate callable service
CSFATKN	ANSI X9.17 key transport key partial notarize callable service
CSFCKI	Clear key import callable service
CSFCKM	Multiple clear key import callable service
CSFCMK	Change master key (TSO panel) utility
CSFCONV	PCF CKSD to ICSF CKDS conversion utility
CSFCPA	Clear PIN generate alternate callable service
CSFCPE	Clear PIN encrypt callable service
CSFCSG	VISA CVV service generate callable service
CSFCSV	VISA CVV service verify callable service
CSFCTT	Cipher text translate callable service
CSFCTT1	Cipher text translate (with ALET) callable service
CSFCVE	Cryptographic variable encipher callable service
CSFCVT	Control vector translate callable service
CSFDCO	Decode callable service
CSFDEC	Decipher callable service
CSFDEC1	Decipher (with ALET) callable service
CSFDKCS	Clear master key entry (TSO panel) utility (PCICC, PCIXCC, CEX2C, or CEX3C)
CSFDKEF	Clear master key entry (TSO panel) utility (CCF)
CSFDKG	Diversified key generate callable service

CSFDKM	Data key import callable service
CSFDKX	Data key export callable service
CSFDSG	Digital signature generate callable service
CSFDSV	Digital signature verify callable service
CSFECO	Encode callable service
CSFEDC	Compatibility service for the PCF CIPHER macro
CSFEMK	Compatibility service for the PCF EMK macro
CSFENC	Encipher callable service
CSFENC1	Encipher (with ALET) callable service
CSFEPG	Encrypted PIN generate callable service
CSFGKC	Compatibility service for the PCF GENKEY macro
CSFIQA	ICSF Query Algorithm callable service
CSFIQF	ICSF Query Facility callable service
CSFKEX	Key export callable service
CSFKGN	Key generate callable service
CSFKGUP	Key generation utility program
CSFKIM	Key import callable service
CSFKPI	Key part import callable service
CSFKRC	Key record create callable service
CSFKRD	Key record delete callable service
CSFKRR	Key record read callable service
CSFKRW	Key record write callable service
CSFKTR	Key translate callable service
CSFKYT	Key test callable service
CSFKYTX	Key test extended callable service
CSFMDG	MDC generate callable service
CSFMDG1	MDC generate (with ALET) callable service
CSFMGN	MAC generate callable service
CSFMGN1	MAC generate (with ALET) callable service
CSFMVR	MAC verify callable service
CSFMVR1	MAC verify (with ALET) callable service
CSFOWH	One-way hash generate and PKCS #11 one-way hash generate callable services
CSFOWH1	One-way hash generate (with ALET) callable service
CSFPCAD	PCICC, PCIXCC, CEX2C, and CEX3C management (TSO panel) utility (activate/deactivate)
CSFPCI	PCI interface callable service
CSFPCU	PIN Change/Unblock callable service

CSFPEX	Prohibit export callable service
CSFPEXX	Prohibit export extended callable service
CSFPGN	Clear PIN generate callable service
CSFPKD	PKA decrypt callable service
CSFPKDR	PKDS reencipher and PKDS activate (TSO panel) utilities
CSFPKE	PKA encrypt callable service
CSFPKG	PKA key generate callable service
CSFPKI	PKA key import callable service
CSFPKRC	PKDS record create callable service
CSFPKRD	PKDS record delete callable service
CSFPKRR	PKDS record read callable service
CSFPKRW	PKDS record write callable service
CSFPKSC	PKSC interface callable service
CSFPKT	PKA key translate callable service
CSFPKTC	PKA key token change callable service
CSFPKX	PKA public key extract callable service
CSFPMCI	Pass phrase master key/KDS initialization (TSO panel) utility
CSFPTR	Encrypted PIN translate callable service
CSFPVR	Encrypted PIN verify callable service
CSFREFR	Refresh CKDS (TSO panel) utility
CSFRENC	Reencipher CKDS (TSO panel) utility
CSFRKD	Retained key delete callable service
CSFRKL	Retained key list callable service
CSFRKX	Remote key export callable service
CSFRNG	Random number generate and PKCS #11 pseudo-random function callable services
CSFRNGL	Random number generate long callable service
CSFRSWS	Administrative control functions (TSO panel) utility (ENABLE)
CSFRTC	Compatibility service for the CUSP or PCF RETKEY macro
CSFSAD	Symmetric Algorithm Decipher
CSFSAD1	Symmetric Algorithm Decipher
CSFSAE	Symmetric Algorithm Encipher
CSFSAE1	Symmetric Algorithm Encipher
CSFSBC	SET block compose callable service
CSFSBD	SET block decompose callable service
CSFSKI	Secure key import callable service
CSFSKM	Multiple secure key import callable service
CSFSKY	Secure messaging for keys callable service

CSFSMK	Set master key (TSO panel) utility
CSFSPN	Secure messaging for PINs callable service
CSFSSWS	Administrative control functions (TSO panel) utility (DISABLE)
CSFSYG	Symmetric key generate callable service
CSFSYI	Symmetric key import callable service
CSFSYX	Symmetric key export callable service
CSFTBC	Trusted block create callable service
CSFTCK	Transform CDMF key callable service
CSFTRV	Transaction validation callable service
CSFUDK	User derived key callable service
CSF1DVK	PKCS #11 derive key callable service.
CSF1DMK	PKCS #11 derive multiple keys callable service.
CSF1HMG	PKCS #11 generate HMAC callable service.
CSF1GKP	PKCS #11 generate key pair callable service.
CSF1GSK	PKCS #11 generate secret key callable service.
CSF1GAV	PKCS #11 get attribute value callable service.
CSF1PKS	PKCS #11 private key sign callable service.
CSF1PKV	PKCS #11 public key verify callable service.
CSF1SKD	PKCS #11 secret key decrypt callable service.
CSF1SKE	PKCS #11 secret key encrypt callable service.
CSF1SAV	PKCS #11 set attribute value callable service.
CSF1TRC	PKCS #11 token record create callable service.
CSF1TRD	PKCS #11 token record delete callable service.
CSF1TRL	PKCS #11 token record list callable service.
CSF1UWK	PKCS #11 unwrap key callable service.
CSF1HMV	PKCS #11 verify HMAC callable service.
CSF1WPK	PKCS #11 wrap key callable service.

Notes:

- 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.
- If you have already started ICSF, you need to refresh the in-storage profiles. See Step 3 on page 50.
- You can specify other operands, such as auditing (AUDIT operand), on the RDEFINE or RALTER commands.
- If the RACF security administrator has activated generic profile checking for the CSFSERV class, you can create generic profiles using the generic characters * and %. This is the same as with any RACF general resource class.

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 CSFCTT UACC(NONE)
```

```
RDEFINE CSFSERV CSFCTT1 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 RACF security administrator to activate the CSFSERV class and refresh the in-storage RACF profiles:

```
SETROPTS CLASSACT(CSFSERV)
```

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

Defining a 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.

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(s) 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 doesn't 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.

Table 3. 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.
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.

Table 3. Key Store Policy controls (continued)

The following Key Store Policy controls:	Consist of the following XFACILIT class resources:	Description:
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 or CSNFSYX) 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 (CSNDSYX or CSNFSYX) callable service. Does not require an active Key Store Policy for CKDS or PKDS. However, if a key token is passed to the callable service instead of a key label, ICSF will, in order to initiate the SAF authorization check, rely on an active Key Store Policy for CKDS.
	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 or CSNFSYX) callable service. Does not require an active Key Store Policy for CKDS or PKDS. However, if a key token is passed to the callable service instead of a key label, ICSF will, in order to initiate the SAF authorization check, rely on an active Key Store Policy for CKDS.

Table 3. Key Store Policy controls (continued)

The following Key Store Policy controls:	Consist of the following XFACILIT class resources:	Description:
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.

For more information on the:

- Key Token Authorization Checking controls, refer to “Enabling access authority checking for key tokens”
- Default Key Label Checking controls, refer to “Determining access to tokens not stored in the CKDS or PKDS” on page 55
- Duplicate Key Token Checking controls, refer to “Enabling duplicate key label checking” on page 56
- Granular Key Label Access controls, refer to “Increasing the level of authority needed to modify key labels” on page 57
- Symmetric Key Label Export controls, refer to “Increasing the level of authority required to export symmetric keys” on page 59
- PKA Key Management Extension control, refer to “Controlling how cryptographic keys can be used” 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 dataset. 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 dataset. 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 Default Key Label Checking control for either the CKDS or PKDS as described in “Enabling duplicate key label checking” on page 56.

If ICSF can not 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 55), 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 4. 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.

Table 4. Key Store Policy controls: The Key Token Authorization Checking controls (continued)

The existence of this resource profile in the XFACILIT class:	Does this:
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 dataset, 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(s) 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 53 for more information. To enable one the Default Key Label Checking controls, create the appropriate profile in the XFACILIT class.

Table 5. 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 6. 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 7. 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 (CSNBKRC) PKDS Record Create (CSNDKRC and CSNFKRC)
Write to a label	CONTROL	Key Part Import (CSNBKPI) Key Record Write (CSNBKRW) PKDS Record Create (CSNDKRC and CSNFKRC) PKDS Record Write (CSNDKRW) PKA Key Generate (CSNDPKG and CSNFPKG) PKA Key Import (CSNDPKI and CSNFPKI) Trusted Block Create (CSNDTBC)
Delete a label	CONTROL	Key Record Delete (CSNBKRD) PKDS Record Delete (CSNDKRD and CSNFKRD) Retained Key Delete (CSNDRKD and CSNFRKD)

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 8. 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 or CSNFSYX) 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 CSNDSYX/CSNFSYX callable service). When the Symmetric Key Label Export control for AES or DES keys is **not** enabled and the CSNDSYX or CSNFSYX 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 CSNDSYX or CSNFSYX 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 CSNDSYX/CSNFSYX callable service; for all other callable services, access to cryptographic keys is checked against profiles in the CSFKEYS class. What's more, 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 CSNDSYX or CSNFSYX callable service 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 9. 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 or CSNFSYX) 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 or CSNFSYX) 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 CSF.XCSFKEY.ENABLE.AES
RDEFINE 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 69.

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)
- Symmetric Key Export (CSNDSYX and CSNFSYX)
- Symmetric Key Import (CSNDSYI and CSNFSYI)

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 69 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 69 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 62

62). 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 (CSNDSYX or CSNFSYX) callable service 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 59 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 CSNDSYX or CSNFSYX 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 10. Keyword settings for symmetric key export using the ICSF segment's SYMEXPORTABLE field

This field/keyword	Specifies:
SYMEXPORTABLE(BYNONE)	The symmetric key can not 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.

Table 10. Keyword settings for symmetric key export using the ICSF segment's SYMEXPORTABLE field (continued)

This field/keyword	Specifies:
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.”
- For more information on using the BYLIST keyword and the SYMEXPORTCERTS field, refer to “Identifying key certificates for symmetric key export” on page 66.
- For more information on the BYANY keyword, refer to “Placing no additional restrictions on symmetric key export” on page 68.

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(s) covered by the profile can not be exported, regardless of a user's access authority to the key. 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, 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(s) 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(s) 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 (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 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”), 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 recreate 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(s) 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 69, 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 65), 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 recreate 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 62, “Restricting asymmetric keys from being used in handshake operations” on page 63, and “Placing restrictions on exporting symmetric keys” on page 64) 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 11. 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 66, 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 62, “Restricting asymmetric keys from being used in handshake operations” on page 63, and “Placing restrictions on exporting symmetric keys” on page 64.

Enabling use of encrypted keys in Symmetric Key Encipher and Symmetric Key Decipher callable services

The Symmetric Key Encipher (CSNBSYE, CSNBSYE1, CSNESYE and CSNESYE1) and Symmetric Key Decipher (CSNBSYD, CSNBSYD1, CSNESYD and CSNESYD1) callable services exploit CP Assist for Cryptographic Functions (CPACF) for improved performance. These services accept AES and DES clear key values and clear key tokens for the key identifier. These services have been enhanced to support encrypted AES and DES key tokens. This support requires the Crypto Express3 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.

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 5. 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.

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 new system or initialize PCICCs, PCIXCCs, CEX2Cs, or CEX3Cs that are brought online after system initialization. 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.

Restriction: If you are running on a system with the Cryptographic Coprocessor Feature, special secure mode must be enabled.

If you plan on sharing your CKDS within your sysplex, refer to Chapter 9, “Running in a Sysplex Environment,” on page 181 for important information. If you have a z9 EC, z9 BC, z10 EC and z10 BC installed, there is an important restriction to consider.

Steps required when 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* and *z/OS Program Directory*.
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*

SAF Protection

The pass phrase initialization utility is primarily protected by the CSFPMCI service name. Only the users authorized to the CSFPMCI service 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 dependant on the crypto processor type and the function(s) of PPINIT that are being utilized.

- CSFOWH
- CSFDKEF (CCF systems only)
- CSFDKCS
- CSFCMK
- CSFMDG
- CSFSMK
- CSFINIT (CCF system only)
- CSFREFR
- CSFPKDR
- CSFRSWS

Running the Pass Phrase Initialization Utility

When you start ICSF, you can use the ICSF panels to run the pass phrase initialization utility. When you access the ICSF panels, the primary menu panel appears. Note that the ICSF FMID appears in the upper left hand corner (it will toggle to the panel identification ID). See Figure 11.

```

HCR7770 ----- Integrated Cryptographic Service Facility -----
OPTION ==> 6

Enter the number of the desired option.

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

Licensed Materials - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1990, 2008. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 11. Selecting the Pass Phrase Initialization Option on the ICSF Primary Menu Panel

Select option 6, PPINIT, and press ENTER to begin the pass phrase initialization utility. The pass phrase panel appropriate for your hardware configuration will appear.

Steps for running PPINIT on a CCF system

The Pass Phrase MK/KDS Initialization panel appears. See Figure 12 on page 75.

```

CSFPMC00 ----- ICSF - Pass Phrase MK/KDS Initialization ---
Command ==>
Enter your pass phrase and the names of the CKDS and PKDS:

Pass Phrase (16 to 64 characters)
==>

CKDS
==>

PKDS
==>

Initialize the CKDS and PKDS? (Y/N) ==> Y
Signature MK = Key Management MK? (Y/N) ==> Y
Initialize new PCICCs only? (Y/N) ==> N

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 12. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel

1. Type the pass phrase and the data set name in the spaces that are provided.

Make sure you save the pass phrase and store it in a secure place.

The CKDS and PKDS names must be valid VSAM data sets.

Notes:

- a. 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.
 - b. If you are reentering master keys when they have been cleared, use the same pass phrase as when you originally entered the keys.
2. Answer the "Initialize the CKDS and PKDS?" question by typing your response in the space following the question.
 - a. If the CKDS and PKDS have not been initialized, type Y.
If you select Y, the CKDS and PKDS names must refer to a valid, uninitialized CKDS and PKDS.
 - b. If this is an existing CKDS and PKDS, type N.
If you select N, the CKDS and PKDS must have already been initialized with the pass phrase initialization utility and the identical pass phrase.
ICSF checks and refreshes the existing CKDS.
 3. Answer the "Signature MK = Key Management MK?" question by typing your response in the space following the question.
 - a. If you have a new system with PCI Cryptographic Coprocessors installed, type Y.
The signature master key and the key management master key will have the same value as the ASYM master key on the PCI Cryptographic Coprocessors. This increases the flexibility in routing services among the cryptographic coprocessors.

- b. If you have previously used pass phrase initialization and you have PKA key tokens that are encrypted under a key management master key that cannot be recreated, type N.
 - c. If none of these two scenarios apply to you, type Y.
4. Answer the "Initialize new PCICCs only?" question by typing your response in the space following the question.
 - a. If you have already initialized your system with the Pass Phrase Initialization utility and now want to initialize new PCI cards, type Y.
 - b. If this is the first time you are running the Pass Phrase Initialization Utility, type N.

```

CSFPMC00 ----- ICSF - Pass Phrase MK/KDS Initialization -----

Enter your pass phrase and the names of the CKDS and PKDS:

Pass Phrase (16 to 64 characters)
====> winnie the pooh and tigger too

CKDS
====> 'CRYPTO.HCRICSF.CKDS'

PKDS
====> 'CRYPTO.HCRICSF.PKDS'

Initialize the CKDS and PKDS? (Y/N) ====> Y
Signature MK = Key Management MK? (Y/N) ====> Y
Initialize new PCICCs only? ====> N

```

Figure 13. Entering Options on the Pass Phrase MK/KDS Initialization Panel

5. Press ENTER to run the utility.

This utility uses the pass phrase, a series of constants, and the MD5 hash algorithm to:

 - Calculate the DES master key and load the new master key registers on the Cryptographic Coprocessor Features with the value.
 - Use the value of the DES master key as the value of the DES-MK key and load the new master key registers on the PCI Cryptographic Coprocessors with the value.
 - Calculate the PKA master keys and set the PKA signature master key register and the PKA key management master key register with these values. If you specified "Y" for the question about making the signature master key equal to the key management master key, then the value calculated for the key management master key will be used for both PKA master keys.
 - Use the value of the PKA signature master key as the value of the ASYM-MK and set the new asymmetric-keys master key registers on the PCI Cryptographic Coprocessors with the value.
 - Set the master key register.
 - Initialize the CKDS or refresh an existing CKDS.
 - Initialize the PKDS.

For details of these calculations, refer to "Pass Phrase Initialization master key calculations" on page 368.

Messages on the bottom half of the panel display the progress of the utility.

6. When the utility has completed successfully, press END to return to the primary menu.

Steps for running PPINIT on a PCIXCC, CEX2C, or CEX3C system

When initializing only new PCIXCCs, CEX2Cs, or CEX3Cs, at least one card must be active and PKA callable services must be enabled.

If you are running on z9 or z10 servers with the Nov. 2008 or later licensed internal code (LIC), see “Steps for running PPINIT with AES master key support” on page 79.

The Pass Phrase MK/CKDS/PKDS Initialization panel appears. See Figure 14.

```
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 asymmetric master keys to all
  coprocessors and initialize the CKDS and the PKDS.
  CKDS ==>
  PKDS ==>

- Reinitialize system - Load the DES and asymmetric master keys to all
  coprocessors and make the specified CKDS and the PKDS the current key data
  sets.
  CKDS ==>
  PKDS ==>

- Add coprocessors - Initialize additional online coprocessors with the
  same DES and asymmetric master keys.

Press ENTER to process.
Press END   to exit to the previous menu.
```

Figure 14. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel

1. Type the pass phrase and the data set names in the spaces that are provided.

Make sure you save the pass phrase and store it in a secure place.

Notes:

- a. 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.
 - b. If you are reentering master keys when they have been cleared, use the same pass phrase as when you originally entered the keys.
2. Select one of the following initialization actions:
 - Select 'Initialize system' if this is the first time you are running the pass phrase initialization utility.

Fill in the CKDS and PKDS fields with the names of two valid VSAM data sets that have not been initialized

- Select 'Reinitialize system' if there is an existing CKDS and PKDS, The CKDS and PKDS must have already been initialized with the pass phrase initialization utility and the identical pass phrase. ICSF checks and refreshes the existing CKDS and PKDS.
- Select 'Add coprocessors' if you have already initialized your system with the Pass Phrase Initialization utility and now want to initialize new PCI cards.

```
CSFPMC10 ----- ICSF - Pass Phrase MK/CKDS/PKDS Initialization ---
Command ==>
Enter your pass phrase (16 to 64 characters)
==> winnie the pooh and tigger too

Select one of the initialization actions then press ENTER to process.

S Initialize system - Load the DES and asymmetric master keys to all
  coprocesors and initialize the CKDS and the PKDS.
  CKDS ==> CRYPTO.HCRICSF.CKDS
  PKDS ==> CRYPTO.HCRICSF.PKDS

_ Reinitialize system - Load the DES and asymmetric master keys to all
  coprocesors and make the specified CKDS and the PKDS the current key data
  sets.
  CKDS ==>
  PKDS ==>

_ Add coprocessors - Initialize additional online coprocessors with the
  same DES and asymmetric master keys.

Press ENTER to process.
Press END   to exit to the previous menu.
```

Figure 15. Entering Options on the Pass Phrase MK/CKDS/PKDS Initialization Panel

3. Press ENTER to run the utility.

This utility uses the pass phrase, a series of constants, and the MD5 hash algorithm to:

- Calculate the DES master key and load the new master key register on the card with the value.
- Calculate the ASYM-MK value and load the new asymmetric-keys master key register on the card with the value.
- Set the master key registers.
- Initialize the CKDS or refresh an existing CKDS.
- Initialize the PKDS.

For details of these calculations, refer to “Pass Phrase Initialization master key calculations” on page 368.

Messages on the bottom half of the panel display the progress of the utility.

4. When the utility has completed successfully, press END to return to the primary menu.
5. If the CKDS has already been initialized and if the DES-MK is valid, this panel appears:

```
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 symmetric master key(s). If  
you proceed with pass phrase initialization, the master key value may change.  
  
If you wish to initialize new coprocessors only, return to the previous panel  
and select add coprocessors only.  
  
To proceed with pass phrase initialization, PKA callable services must be  
disabled. Use the Administrative Control Function utility to disable PKA  
callable services.  
  
Press ENTER to proceed with pass phrase initialization.  
Press END to exit to the previous menu.
```

Figure 16. Pass Phrase MK/CKDS/PKDS Initialization Panel

This prevents you from making a mistake and changing a system that is already operational.

Steps for running PPINIT with AES master key support

|
|

When initializing only new CEX2Cs or CEX3Cs, at least one card must be active and PKA callable services must be enabled.

If you are running on z9 or z10 servers with the Nov. 2008 or later licensed internal code (LIC), the Pass Phrase MK/KDS Initialization panel appears. See Figure 17 on page 80.

```

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 asymmetric master keys to all
  coprocessors and initialize the CKDS and the PKDS.
  CKDS ===>
  PKDS ===>

_ Reinitialize system - Load the AES, DES and asymmetric master keys to all
  coprocessors and make the specified CKDS and the PKDS the current key data
  sets.
  CKDS ===>
  PKDS ===>

_ Add coprocessors - Initialize additional online coprocessors with the
  same DES and asymmetric 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.

```

Figure 17. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel

1. Type the pass phrase and the data set names in the spaces that are provided.
Make sure you save the pass phrase and store it in a secure location.

Notes:

- a. 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.
 - b. If you are reentering master keys when they have been cleared, use the same pass phrase as when you originally entered the keys.
2. Select one of the following initialization actions:
 - Select 'Initialize system' if this is the first time you are running the pass phrase initialization utility.
Save the pass phrase in a secure place.
The CKDS and PKDS names must refer to a valid CKDS and PKDS in your system that have not been initialized.
 - Select 'Reinitialize system' if there is an existing CKDS and PKDS,
The CKDS and PKDS must have already been initialized with the pass phrase initialization utility and the identical pass phrase.
ICSF checks and refreshes the existing CKDS and PKDS.
 - Select 'Add coprocessors' if you have previously initialized your system with the Pass Phrase Initialization utility and now want to initialize new PCI cards.

- Select 'Add AES-MK' if you want to add secure key AES support to a system previously initialized with the utility. This selection updates the active CKDS.

```

CSFPMC30 ----- ICSF - Pass Phrase MK/CKDS/PKDS Initialization ---
Command ==>
Enter your pass phrase (16 to 64 characters)
==> winnie the pooh and tigger too

Select one of the initialization actions then press ENTER to process.

S Initialize system - Load the AES, DES and asymmetric master keys to all
  coprocessors and initialize the CKDS and the PKDS.
  CKDS ==> CRYPTO.HCRICSF.CKDS
  PKDS ==> CRYPTO.HCRICSF.CKDS

- Reinitialize system - Load the AES, DES and asymmetric master keys to all
  coprocessors and make the specified CKDS and the PKDS the current key data
  sets.
  CKDS ==>
  PKDS ==>

- Add coprocessors - Initialize additional online coprocessors with the
  same DES and asymmetric 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.

```

Figure 18. Entering Options on the Pass Phrase MK/CKDS/PKDS Initialization Panel

3. Press ENTER to run the utility.

This utility uses the pass phrase, a series of constants, and the MD5 hash algorithm to:

- Calculate the DES and AES master key values and loads the new master key register on the CEX2C or CEX3C with the value.
- Calculate the ASYM-MK value and load the new asymmetric-keys master key register on the CEX2C or CEX3C with the value.
- Set the master key registers.
- Initialize the CKDS or refresh an existing CKDS.
- Initialize the PKDS.

For details of these calculations, refer to “Pass Phrase Initialization master key calculations” on page 368.

Messages on the bottom half of the panel display the progress of the utility.

4. When the utility has completed successfully, press END to return to the primary menu.
5. If the CKDS has been initialized and if either the DES or AES master key is valid, this panel appears:

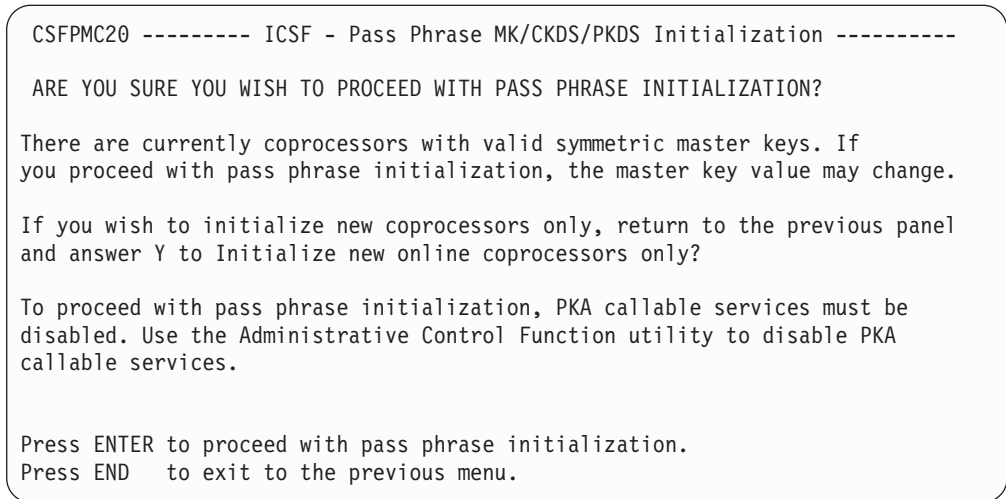


Figure 19. Pass Phrase MK/CKDS/PKDS Initialization Panel

This prevents you from making a mistake and changing a system that is already operational.

Steps for adding a PCICC after first time Pass Phrase Initialization

The pass phrase initialization utility can be used to initialize PCI Cryptographic Coprocessors after system initialization. The procedure is to re-run the Pass Phrase Initialization Utility.

Note: Special Secure Mode is not required when adding PCICCs after first time pass phrase initialization.

The step-by-step procedure is:

1. Run the Pass Phrase Initialization Utility.
Access the primary menu panel.

```

CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 6

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

      5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 20. Selecting the Pass Phrase Initialization Option on the ICSF Primary Menu Panel

2. Select option 6, PPINIT, and press ENTER to begin the pass phrase initialization utility.

The Pass Phrase MK/KDS Initialization panel appears. See Figure 21.

```

CSFPMC00 ----- ICSF - Pass Phrase MK/KDS Initialization ---
Command ==>
Enter your pass phrase and the names of the CKDS and PKDS:

Pass Phrase (16 to 64 characters)
==>

CKDS
==>

PKDS
==>

Initialize the CKDS and PKDS? (Y/N) ==>
Signature MK = Key Management MK? (Y/N) ==>
Initialize new PCICCs only? (Y/N) ==>

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 21. ICSF Pass Phrase MK/KDS Initialization Panel

3. Type the pass phrase and the data set name in the spaces that are provided. The CKDS and PKDS names must be the current, active CKDS and PKDS.

Note: The same pass phrase will always produce the same master key values. Because you are reentering master keys, you must use the same pass

phrase as when you originally entered the keys. You should have saved the pass phrase in a secure place when you entered the master keys previously.

4. The "Initialize the CKDS and PKDS?" and "Signature MK = Key Management MK?" questions are ignored.
5. Answer the "Initialize new PCICCs only" question by typing your response in the space following the question. Your response should be Y.

```
CSFPMC00 ----- ICSF - Pass Phrase MK/KDS Initialization -----  
  
Enter your pass phrase and the names of the CKDS and PKDS:  
  
Pass Phrase (16 to 64 characters)  
====> winnie the pooh and tigger too  
  
CKDS  
====> 'CRYPTO.HCRICSF.CKDS'  
  
PKDS  
====> CRYPTO.HCRICSF.PKDS  
  
Initialize the CKDS and PKDS? (Y/N) ====> N  
Signature MK = Key Management MK? (Y/N) ====> Y  
Initialize new PCICCs only? ====> Y
```

Figure 22. Entering Options on the Pass Phrase MK/KDS Initialization Panel

6. Press ENTER to run the utility.
For details of these calculations, refer to "Pass Phrase Initialization master key calculations" on page 368.
Messages on the bottom half of the panel display the progress of the utility.
7. When the utility has completed successfully, press END to return to the primary menu.

Steps for adding a PCIXCC, CEX2C, or CEX3C after first time Pass Phrase Initialization

The pass phrase initialization utility can be used to initialize PCIXCCs, CEX2Cs, or CEX3Cs after system initialization. The procedure is to rerun the Pass Phrase Initialization Utility.

When using PPINIT with a system where coprocessors have been initialized with PPINIT (the CKDS/PKDS are initialized), keep in mind:

- When the CKDS and PKDS were initialized with the same pass phrase, the 'Reinitialize system' option will only process active coprocessors. Online coprocessors will not become active. To initialize online coprocessors in this scenario, use the 'Add coprocessors' option.
- When the CKDS and PKDS were initialized with a different pass phrase, the 'Reinitialize system' option will process all coprocessors (active and online) and all coprocessors will become active.

The step-by-step procedure is:

1. Run the Pass Phrase Initialization Utility.
Access the primary menu panel.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----  
OPTION ==> 6
```

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Press ENTER to go to the selected option.

Press END to exit to the previous menu.

Figure 23. Selecting the Pass Phrase Initialization Option on the ICSF Primary Menu Panel

2. Select option 6, PPINIT, and press ENTER to begin the pass phrase initialization utility.

The Pass Phrase MK/CKDS/PKDS Initialization panel appears. See Figure 24 on page 86.

Note: Panel CSFPMC30 appears if you are running on a IBM System z10 Enterprise Class or IBM System z10 Business Class with the Nov. 2008 or later licensed internal code (LIC).

```

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 asymmetric master keys to all
  coprocessors and initialize the CKDS and the PKDS.
  CKDS ===>
  PKDS ===>

_ Reinitialize system - Load the DES and asymmetric master keys to all
  coprocessors and make the specified CKDS and the PKDS the current key data
  sets.
  CKDS ===>
  PKDS ===>

_ Add coprocessors - Initialize additional online coprocessors with the
  same DES and asymmetric master keys.

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 24. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel

3. Type the pass phrase and the data set name in the spaces that are provided. Refer to the example in Figure 25 on page 87.
The CKDS and PKDS names must be the current, active CKDS and PKDS.

Note: The same pass phrase will always produce the same master key values. Because you are reentering master keys, you must use the same pass phrase as when you originally entered the keys. You should have saved the pass phrase in a secure place when you entered the master keys previously.

4. Select the 'Add coprocessors' action.

```

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 asymmetric master keys to all
  coprocessors and initialize the CKDS and the PKDS.
  CKDS ==>
  PKDS ==>

_ Reinitialize system - Load the DES and asymmetric master keys to all
  coprocessors and make the specified CKDS and the PKDS the current key data
  sets.
  CKDS ==>
  PKDS ==>

s Add coprocessors - Initialize additional online coprocessors with the
  same DES and asymmetric master keys.

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 25. Entering Options on the Pass Phrase MK/CKDS/PKDS Initialization Panel

5. Press ENTER to run the utility.

For details of these calculations, refer to “Pass Phrase Initialization master key calculations” on page 368.

Messages on the bottom half of the panel display the progress of the utility.

6. When the utility has completed successfully, press END to return to the primary menu.

Migrating to a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC server

If you are migrating to a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC server from a CCF system, follow this procedure.

Assumptions are:

1. You used PPINIT to initialize your CKDS and PKDS.
2. You have not changed your master key since running PPINIT.
3. You are migrating to a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC from a CCF system and using the same CKDS and PKDS from assumption number 1.

Note: If you are planning to use a PKDS and CKDS, you must set the master keys; a CEX2C or CEX3C is required for this task. The CEX2C is supported on the z9 EC, z9 BC, z10 EC, and z10 BC. The CEX3C is supported on the z10 EC and z10 BC. Ensure that you have at least one CEX2C or CEX3C configured on your server.

The procedure is as follows:

- Access the primary menu panel and select option 6, PPINIT. The Pass Phrase MK/KDS Initialization panel appears.

Note: Panel CSFPMC30 appears if you are running on a IBM System z10 Enterprise Class or IBM System z10 Business Class with the Nov. 2008 or later licensed internal code (LIC).

- Select the 'Reinitialize system' action and enter the same pass phrase from assumption number 1 on page 87 and the same CKDS and PKDS.

```
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 asymmetric master keys to all
  coprocessors and initialize the CKDS and the PKDS.
  CKDS ==>
  PKDS ==>

s Reinitialize system - Load the DES and asymmetric master keys to all
  coprocessors and make the specified CKDS and the PKDS the current key data
  sets.
  CKDS ==>
  PKDS ==>

_ Add coprocessors - Initialize additional online coprocessors with the
  same DES and asymmetric master keys.

Press ENTER to process.
Press END   to exit to the previous menu.
```

Figure 26. ICSF Pass Phrase MK/CKDS/PKDS Initialization Panel

- Press ENTER to run the utility
- When the utility has completed successfully, press END to return to the primary menu.

PPINIT Recovery

If you are unsuccessful using the pass phrase initialization, you should follow one of these procedures. Your recovery steps will vary as they are dependent on your hardware configuration.

Steps recovering with a CCF (with or without a PCICC)

If your panel message returns NOT SUCCESSFUL or PPINIT fails to complete, try to:

1. Delete and reallocate the CKDS
2. Delete and reallocate the PKDS
3. Go to the ICSF Coprocessor Management Panel to list the coprocessors and their status:

```
CSFCMP00 ----- ICSF Coprocessor Management -----
COMMAND ==>
```

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

COPROCESSOR	MODULE ID/SERIAL NUMBER	STATUS
-----	-----	-----
- A06		ACTIVE
- A07		ACTIVE
s C0	E589C396944007A6 5D40369997A386F4	ONLINE
s C1	79BF0AA3D2387960 0367DC04533125FF	ACTIVE
- P00	41-00YE1	ONLINE
- P01	41-00K11	ONLINE
- P02	41-0A355	ONLINE
- P03	41-0BA3F	ONLINE
- P04	41-0RT2T	DEACTIVATED
- P05	41-00342	DISABLED

4. Make sure that these registers are EMPTY (for C0 and C1): DES new master key register, the current signature master key register (SMK) and the PKA key management master key register (KMMK). On a z900, you should see this:

```

CSFCMP10 ----- ICSF - Coprocessor Hardware Status -----
OPTION ==>

CRYPTO DOMAIN: 0

REGISTER STATUS          COPROCESSOR C0          COPROCESSOR C1
                          More: +
Crypto Serial Number or  : E589C396944007A6  79BF0AA3D2387960
Module Id                : 5D40C39997A396F0  0367DC04533125FF
Status                   : ONLINE           ACTIVE
DES/Symmetric-Keys Master Key
New master key register  : EMPTY           EMPTY
Verification pattern     :
Hash pattern             :
                          :
Old master key register  : EMPTY           EMPTY
Verification pattern     :
Hash pattern             :
                          :
Current master key register : VALID           VALID
Verification pattern     : CA6B408A02371B1D  261AAB8A02371705
Hash pattern             : 41DF774FF81547D0  562A5202F8154331
                          : 090ABC4539727511  4093990AB1202451
PKA Signature/Asymmetric-Keys Master Key
New master key register  : N/A           N/A
Hash pattern             :
                          :
Old master key register  : N/A           N/A
Hash pattern             :
                          :
Current master key register : EMPTY           EMPTY
Hash pattern             :
                          :
PKA Key Management Master Key register
Hash pattern             : EMPTY           EMPTY
                          :
Special Secure Mode      : Enabled       Enabled
Environment Control Mask : FBFEFCF0  FBFEFCF0
Crypto Configuration Control : EF569412CD91AB78 EF569412CD91AB78
                          : 1F25A78BC8ED77A  1F25A78BC8ED77A

Press ENTER to refresh the hardware status display.
Press END to exit to the previous menu.

```

Figure 27. Coprocessor Hardware Status Panel

5. If the registers are not EMPTY, go to “Entering master key parts” on page 93. Reset the registers that are not EMPTY. Be sure to check both C0 and C1.
6. If you have one or more PCICCs, there is no checking to be done.
7. Rerun PPINIT.

Steps recovering with a PCIXCC, CEX2C, or CEX3C

If your panel message returns NOT SUCCESSFUL or PPINIT fails to complete, try this:

1. Delete and reallocate the CKDS
2. Delete and reallocate the PKDS
3. Rerun PPINIT.

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 6. Managing Master Keys - CCF and PCICC

This topic describes how to use the master key entry panels to enter master keys in the Cryptographic Coprocessor Feature and PCI Cryptographic Coprocessor.

You can have up to two Cryptographic Coprocessor Features on each IBM @server zSeries 800 and IBM @server zSeries 900. Each Cryptographic Coprocessor Feature is capable of performing cryptographic functions and holding the master keys within a secure boundary.

You can have multiple PCI Cryptographic Coprocessors and PCI Cryptographic Accelerators on these servers. There can be a total of 16.

Each PCI Cryptographic Coprocessor is capable of performing cryptographic functions and holding the master keys within a secure boundary. The PCI Cryptographic Coprocessors work in conjunction with the Cryptographic Coprocessor Features on your server.

Restriction: The CCF and PCICC are only available on the IBM @server zSeries 800 and the IBM @server zSeries 900 processors.

Requests for cryptographic services are routed to either the PCICC or CCF, depending on key types specified in the request. In order for these two types of cryptographic coprocessors to work together, you need to install the same master key values for each coprocessor.

Note: The PCI Cryptographic Accelerators improve private key decryption performance. They do not require setting of master keys.

Entering master key parts

You can use the Master Key Entry panels to enter master key parts in the clear. 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 how you get the master key parts, **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.

When you enter the PKA master keys and the asymmetric-keys master key (ASYM-MK) the first time, the PKA callable services are initially disabled. Once you have entered the PKA master keys and the ASYM-MK, you must enable the PKA callable services for these services to work. When you change the PKA master keys and the ASYM-MK, you need to disable the PKA callable services. To enable and disable the PKA callable services refer to “Steps for enabling and disabling PKA services” on page 124.

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 101.

To begin master key entry by generating random numbers for the key parts, continue with “Generating master key data for master key entry” on page 94.

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.

A DES master key is 16 bytes long. A symmetric-keys master key (SYM-MK) is 24 bytes long. ICSF enforces the SYM-MK to be 16 bytes long. ICSF defines these master keys by exclusive ORing two or more key parts. Each of the master key parts is also 16 bytes long. To enter either a DES master key or a SYM-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 when entering the final key part.

Note: The combined DES master key is forced to have odd parity, but the parity of the individual key parts can be odd, even or mixed. We refer to even or mixed parity keys as non-odd parity keys.

Attention: The PCICC will not allow certain 'weak' keys as master keys. The list of weak keys are documented in Appendix F, "Questionable (Weak) Keys," on page 381. If you have an existing CCF installed with a weak master key, you can not install that master key in the PCICC. You must change the CCF master keys and load those same master keys in the PCICCs.

PKA master keys and the ASYM-MKs are each 24 bytes long. ICSF defines these master keys by exclusive ORing two or more key parts. Each of the PKA master key parts is also 24 bytes long.

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.

Note: It is recommended that you enter the same key value for the SMK and KMMK of the Cryptographic Coprocessor Feature and the ASYM-MK of the PCI Cryptographic Coprocessor Feature. This will allow ICSF flexibility in workload balancing.

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.

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 master key or SYM-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 PKA master key (SMK, KMMK or ASYM-MK), the coprocessor calculates a sixteen-byte hash pattern.

When the verification and hash patterns can be calculated, the DES master key must have been set.

The ICSF Master Key Entry panel displays the verification pattern or hash pattern. Check the displayed verification pattern against the optional verification pattern you may have generated at the time you generated the DES or SYM-MK master key parts and the checksum. Check the displayed hash pattern against the optional hash pattern that you may have generated at the same time you generated the PKA master key part and the checksum. The verification pattern or hash pattern checks whether you entered the key part correctly, and whether you entered the correct key type.

ICSF displays a verification and hash pattern for each DES master key part. It also displays a verification and hash pattern for the DES master key when you enter all the key parts. If the verification and hash patterns are the same, you have entered the key part correctly. Likewise, in addition to displaying a hash pattern for each PKA master key part, ICSF also displays a hash pattern for the PKA master key when you enter all the key parts. If the hash patterns are the same, you have entered the key part correctly.

Note: Keys stored in the CKDS are enciphered under the DES master key. The master key verification pattern is stored in the CKDS header record. Checking the verification pattern is optional; it is not required for key entry.

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*.

Note: ICSF must be initialized with a DES master key to use the random number generate callable service or the Random Number Generator panel.

These topics describe using the ICSF utilities to generate key parts, checksums, verification patterns, and hash patterns.

Steps for generating key parts using ICSF utilities

1. Access ICSF utilities by choosing option 5, UTILITY, on the Primary Menu panel, as shown in Figure 28 on page 96.

```

CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 5

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 28. Selecting the Utility Option on the ICSF Primary Menu Panel

The Utilities panel appears. See Figure 29. You use the RANDOM and CHECKSUM options to generate random numbers, checksums, and verification patterns for master key management.

```

CSFUTL00 ----- ICSF - Utilities -----
OPTION ==> 3

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 29. ICSF Utilities Panel

2. Choose option 3, RANDOM, to access the Random Number Generator panel, shown in Figure 30 on page 97.

```
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
```

Figure 30. ICSF Random Number Generator Panel

3. To select the parity of the random numbers, enter ODD, EVEN, or RANDOM next to Parity Option and press ENTER.

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 PKA 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.

Note: The third random number is only for PKA master keys. It is not used for DES master keys or operational keys.

```
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 : 0000000000000000 Random Number 4
```

Figure 31. 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.

Important: 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 ICSF 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. The random number generator and the hash and verification pattern, however, do not work until ICSF has been initialized with a valid master key.

Note: The use of these utility panels 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 365 for a description of the checksum algorithm. See “Algorithm for calculating a verification pattern” on page 367 for a description of the algorithm for the verification pattern. See “The MDC-4 Algorithm for Generating Hash Patterns” on page 368 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 a 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 32.

```
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 32. Selecting the Checksum Option on the ICSF Utilities Panel

The Checksum and Verification and Hash Pattern panel appears. See Figure 33 on page 99.

```

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 and PKA keys only)
               ==> 0000000000000000  Input key value 4 (AES keys only)

Checksum      : 00                                Check digit for key value
Key Part VP   : 0000000000000000  Verification Pattern
Key Part HP   : 0000000000000000  Hash Pattern
               : 0000000000000000

```

Figure 33. ICSF Checksum and Verification and Hash Pattern Panel

If you accessed the Random Number Generator panel before 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 later specify in the Master Key Entry panels, make sure you record the numbers.
3. In the Key Type field, specify either:
 - MASTER to generate a checksum and hash and verification pattern for a DES master key part.
 - PKAMSTR to generate a checksum and hash pattern for a PKA master key part.

If you leave the Key Type field blank and press ENTER, the Key Type Selection panel appears. See Figure 34.

```

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
  EXPORTER       Export key encrypting key
  IMP-PKA        Limited authority importer key
  IMPORTER       Import key encrypting key
  IPINENC        Input PIN encrypting key
  s MASTER       DES master key
  OPINENC        Output PIN encrypting key
  PINGEN         PIN generation key
  PINVER         PIN verification key
  PKAMSTR        PKA master key
***** BOTTOM OF DATA *****

```

Figure 34. Key Type Selection Panel Displayed During Hardware Key Entry

4. Type 'S' to the left of the MASTER key type, and press ENTER to return to the Checksum and Verification Pattern panel as shown in Figure 35.

In this example, we have selected the DES master key.

```
CSFMKV00 ----- ICSF - Checksum and Verification and Hash Pattern ---
COMMAND ==>

Enter data below:

Key Type      ==> MASTER                (Selection panel displayed if blank)

Key Value     ==> 51ED9CFA90716CFB      Input key value 1
               ==> 58403BFA02BD13E8      Input key value 2
               ==> 9B28AEFA8C47760F      Input key value 3 (AES and PKA keys only)
               ==> 0000000000000000      Input key value 4 (AES keys only)

Checksum      : 00                      Check digit for key part
Key Part VP   : 0000000000000000      Verification Pattern
Key Part HP   : 0000000000000000      Hash Pattern
               : 0000000000000000
```

Figure 35. ICSF Checksum and Verification Pattern Panel

5. On the Checksum and Verification Pattern panel, press ENTER.

ICSF calculates the checksum, verification pattern, and hash pattern for the key part segments and displays them on the panel as shown in Figure 36. Since a DES master key was selected for this example, the key part last segment was not used in the calculations. The key part last field is zeroed out on the panel. For a PKA master key, ICSF uses all three key part segments to calculate the checksum and hash pattern.

```
CSFMKV00 ----- ICSF - Checksum and Verification and Hash Pattern ---
COMMAND ==>

Enter data below:

Key Type      ==> MASTER                (Selection panel displayed if blank)

Key Value     ==> 51ED9CFA90716CFB      Input key value 1
               ==> 58403BFA02BD13E8      Input key value 2
               ==> 0000000000000000      Input key value 3 (AES and PKA keys only)
               ==> 0000000000000000      Input key value 4 (AES keys only)

Checksum      : 40                      Check digit for key part
Key Part VP   : 0CCE190A635A6C89      Verification Pattern
Key Part HP   : EA58E51179754FB7      Hash Pattern
               : C102957465CE479E
```

Figure 36. Checksum, Verification Pattern, and Hash Pattern Calculated for a DES Master Key Part

6. Record the checksum, verification pattern, and hash pattern.

Save these values in a secure place along with the key part values in case of a tamper. If the Cryptographic Coprocessor Feature detects tampering, it clears the master key, and you have to reenter the same master key again.

7. Press END to return to the Utilities panel.

8. 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 104.
- If you have generated a final master key part, continue with “Steps for entering the final key part” on page 105.

Steps for entering the first master key part

Use the Master Key Entry panels to enter each key part.

Important: When entering 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 parts 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 when generating another key part.

To enter master key parts:

1. Select option 1, COPROCESSOR MGMT, on the ICSF Primary menu, as shown in Figure 37, and press ENTER.

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

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 37. Selecting the Coprocessor Management option on the primary menu panel

The ICSF Coprocessor Management panel appears (Figure 38 on page 102).

2. Select the coprocessor(s) 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 ONLINE. When the master keys are set, status will be ACTIVE.

```
CSFCMP00 ----- ICSF Coprocessor Management -----
COMMAND ==>

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                STATUS
-----
- A06                                     ACTIVE
- A07                                     ACTIVE
E C0          E589C396944007A6 5D40369997A386F4      ONLINE
E C1          0AA379BFD2387960 0367DC04533125FF      ONLINE
E P00         41-00YE1                                ONLINE
E P01         41-00K11                                ONLINE
E P02         41-0A355                                ONLINE
E P03         41-0BA3F                                ONLINE
- P04         41-0RT2T                                DEACTIVATED
- P05         41-00342                                DISABLED
```

Figure 38. Selecting the coprocessor on the Coprocessor Management Panel

3. The ICSF Master Key Entry panel appears. See Figure 39.

```
CSFDKE10----- ICSF - Master Key Entry -----
COMMAND ==>

CCF DES/PCICC SYM-MK new master key register      : EMPTY
CCF Signature/PCICC ASYM-MK master key register    : EMPTY
CCF Key management master key register             : EMPTY

Specify information below
Key Type ==> ____ (DES, SMK, KMMK, ALL-PKA)

Part      ==> ____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 40

Key Value ==> 51ED9CFA90716CFB
           ==> 58403BFA02BD13E8
           ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)

Press ENTER to process.
Press END   to exit to the previous menu.
```

Figure 39. 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 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 random number generate).
- 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 Cryptographic Coprocessor Feature calculated, the key part is accepted. The message at the top of the panel states KEY PART LOADED, as shown in Figure 40. 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.

```

CSFDKE10 ----- ICSF - Master Key Entry --- KEY PART LOADED
COMMAND ==>

          CCF DES/PCICC SYM-MK new master key register      : PART FULL
          CCF Signature/PCICC ASYM-MK master key register   : EMPTY
          CCF Key management master key register            : EMPTY

Specify information below
Key Type  ==> DES          (DES, SMK, KMMK, ALL-PKA)

Part      ==> FIRST       (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
          ==> 0000000000000000
          ==> 0000000000000000 (SMK, KMMK and ALL-PKA 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 40. 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 95 if you are using the ICSF utilities to generate random numbers for key values.
- “Steps for entering intermediate key parts” 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 when you enter the first key part and when you enter 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(s) 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 41).

```
CSFDKE10 ----- ICSF - Master Key Entry -----
COMMAND ==>

                CCF DES/PCICC SYM-MK new master key register      : PART FULL
                CCF Signature/PCICC ASYM-MK master key register   : EMPTY
                CCF Key management master key register            : EMPTY

Specify information below
Key Type ==>  ____ (DES, SMK, KMMK, ALL-PKA)

Part      ==>  _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==>  42

Key Value ==>  4C2269A1008A754D
              ==>  B7642C135F68329A
              ==>  0000000000000000 (SMK, KMMK and ALL-PKA only)
```

Figure 41. 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 generate).
 - 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 Cryptographic Coprocessor Feature calculated, the key part is accepted.

The message at the top of the panel states KEY PART LOADED, as shown in Figure 42. 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.

```
CSFDKE10 ----- ICSF - Master Key Entry -----KEY PART LOADED
COMMAND ==>

          CCF DES/PCICC SYM-MK new master key register      : PART FULL
          CCF Signature/PCICC ASYM-MK master key register   : EMPTY
          CCF Key management master key register            : EMPTY

Specify information below
Key Type ==> DES      (DES, SMK, KMMK, ALL-PKA)

Part      ==> MIDDLE (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
          ==> 0000000000000000
          ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)

Entered key part VP: 8D8A000BE067EBF7 HP: 9D92F343479D77F2 229FD4CDB49C2679

          (Record and secure these patterns)
```

Figure 42. 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 95 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

When 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(s) to be processed by entering an 'E' on the Coprocessor Management panel.
3. When pressing ENTER, the Master Key Entry panel appears.

```

CSFDKE10 ----- ICSF - Master Key Entry -----
COMMAND ===>

                CCF DES/PCICC SYM-MK new master key register      : PART FULL
                CCF Signature/PCICC ASYM-MK master key register   : EMPTY
                CCF Key management master key register            : EMPTY

Specify information below
Key Type  ===>  _____ (DES, SMK, KMMK, ALL-PKA)

Part      ===>  _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ===>  4A

Key Value ===>  8697ACDC2431BABA
               ===>  CE369D24680E9753
               ===>  0000000000000000 (SMK, KMMK and ALL-PKA only)

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 43. 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 generate).
 - 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 Cryptographic Coprocessor Feature calculated, the key part is accepted. The message at the top of the panel states KEY PART LOADED, as shown in Figure 44 on page 107. 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.

```

CSFDKE10 ----- ICSF - Master Key Entry ----- KEY PART LOADED
COMMAND ==>

          CCF DES/PCICC SYM-MK new master key register      : FULL
          CCF Signature/PCICC ASYM-MK master key register   : EMPTY
          CCF Key management master key register            : EMPTY

Specify information below
Key Type ==> DES          (DES, SMK, KMMK, ALL-PKA)

Part      ==> FINAL      (RESET, FIRST, MIDDLE, FINAL)

Checksum ==> 00

Key Value ==> 0000000000000000
          ==> 0000000000000000
          ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)

Entered key part VP: 8D8A000BE067EBF7 HP: 9D92F343479D77F2 229FD4CDB49C2679
Master Key      VP: 8F887096A8D4922C HP: 4C887096A8D4922B 33387096A8D4922B
                (Record and secure these patterns)

```

Figure 44. 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, since 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 108.
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, ICSF places the verification pattern for the DES master key into the CKDS header record.

When you have entered the master key parts 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 master key and SYM-MK for the first time, initialize the CKDS with the DES master key. For a description of the process of initializing the CKDS with the DES master key on your system, see “Initializing the CKDS and PKDS at First-Time Startup” on page 110.
- If you are defining a DES master key after it was cleared, set the DES master key to make it active. For a description of the process of recovering from tampering, see “Reentering master keys when they have been cleared” on page 117.
- If you are changing a DES master key, reencipher the CKDS under the new DES master key and make it active. For a description of the process of changing a DES master key, see “Steps for changing master keys” on page 119.
- If you are changing the PKA Master Key, see “Steps for changing PKA master keys” on page 125.

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.

Note: If you are working on a CCF, when you enter the first key part, your old master key is lost, even if you restart the process.

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.

```
CSFDKE10 ----- ICSF - Master Key Entry -----
COMMAND ==>

                CCF DES/PCICC SYM-MK new master key register      : PART FULL
                CCF Signature/PCICC ASYM-MK master key register  : EMPTY
                CCF Key management master key register           : EMPTY

Specify information below
Key Type ==>  DES                (DES, SMK, KMMK, ALL-PKA)

Part      ==>  RESET_            (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==>  00

Key Value ==>  0000000000000000
              ==>  0000000000000000
              ==>  0000000000000000 (SMK, KMMK and ALL-PKA only)
```

Figure 45. Selecting Reset on the Master Key Entry Panel

3. Press ENTER.
The Restart Key Entry Process panel appears. See Figure 46 on page 109. This panel confirms your request to restart the key entry process.

```

CSFDKE40 ----- ICSF - Restart Key Entry Process -----
COMMAND ==>

ARE YOU SURE YOU WISH TO RESTART THE KEY ENTRY PROCESS?

Restarting the process will clear the DES master key register.

WARNING: Resetting the KMMK or SMK will invalidate any private
         internal key tokens in the PKDS.

Press ENTER to confirm restart request
Press END   to cancel restart request

```

Figure 46. Confirm Restart Request Panel

Note: If you are restarting the key entry process for one or all of the PKA master keys, the panel message will differ. ICSF substitutes either 'KMMK register', 'SMK register' or 'ALL-PKA register' for 'the DES master key register' phrase in the panel message.

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 47. This indicates that the register has been cleared.

```

CSFDKE10 ----- ICSF - Master Key Entry -----
COMMAND ==>

CCF DES/PCICC SYM-MK new master key register      : EMPTY
CCF Signature/PCICC ASYM-MK master key register    : EMPTY
CCF Key management master key register             : EMPTY

Specify information below
Key Type ==> ____ (DES, SMK, KMMK, ALL-PKA)

Part      ==> _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
           ==> 0000000000000000
           ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)

```

Figure 47. 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.

Initializing the CKDS and PKDS at First-Time Startup

The first time you start ICSF, you must:

- Create a cryptographic key data set (CKDS)
- Create a PKA key data set (PKDS)
- Enter a DES new master key into the Cryptographic Coprocessor Feature
- Enter a new SYM-MK into each PCI Cryptographic Coprocessor, if you have PCICCs in your environment
- Initialize the CKDS
- Enter PKA Key Management and Signature master keys into the Cryptographic Coprocessor Feature
- Enter a new ASYM-MK into each PCI Cryptographic Coprocessor, if you have PCICCs in your environment
- Initialize the PKDS

Note: Once these tasks are completed, you should enable PKA callable services and PKDS read and write access.

When you initialize the CKDS, ICSF creates a header record for the CKDS, and sets the DES master key. Keys stored in the CKDS are enciphered under the DES master key.

CKDS

When you define the DES master key and initialize a CKDS, you can generate or enter any additional system keys you need to perform cryptographic functions.

If you are running on a IBM @server zSeries 990, and wish to share your CKDS and PKDS with an IBM @server zSeries 800 or IBM @server zSeries 900 (which might be your disaster recovery site), the CKDS and PKDS should be initialized on the IBM @server zSeries 800 or IBM @server zSeries 900.

There are four different types of system keys you can install in the CKDS:

- Required SYSTEM keys are automatically generated when you first initialize the CKDS. These include the MAC and MACVER keys that ICSF uses to generate and validate the MAC code in each CKDS record.
- NOCV-enablement keys are required for NOCV IMPORTERs and EXPORTERs. The NOCV-enablement system keys are used to twist on and twist off the CVs on external tokens during key import and key export. This allows ICSF to communicate with systems that do not use control vectors.
- ANSI system keys are required for almost all ANSI services to perform the notarization and offset that are required by ANSI X9.17.
- ESYS, or enhanced system keys, are used only in Symmetric Key Export service.

For information on system keys, see “Entering system keys into the cryptographic key data set (CKDS)” on page 29.

If running in a sysplex, see Chapter 9, “Running in a Sysplex Environment,” on page 181.

Steps for initializing a CKDS

You have to initialize a CKDS only the first time you start ICSF on a system. 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. At any time, you can read a different disk copy into storage. For information about how to read a disk copy into storage, see “Refreshing the CKDS at any time” on page 115. For information about initializing a CKDS in a sysplex environment, see Chapter 9, “Running in a Sysplex Environment,” on page 181.

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 101. For a description of how to use the TKE workstation to enter the master key, refer to *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

To initialize the CKDS:

1. Return to the Primary Menu panel by pressing END from the Master Key Entry panel.
2. Select Option 2, MASTER KEY, on the Primary Menu panel as shown in Figure 48.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 2

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 48. Selecting the Master Key option on the primary menu panel

The Master Key Management panel appears. See Figure 49 on page 112.

```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 1

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                   - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS         - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK               - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS         - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS         - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS           - Activate an updated PKA Cryptographic Key Data Set

```

Figure 49. ICSF Master Key Management Panel

3. Select option 1, INIT/REFRESH CKDS and the Initialize a CKDS panel appears. See Figure 50.

```

CSFCKD00 ----- ICSF - Initialize a CKDS -----
COMMAND ==> 1

Enter the number of the desired option.

 1 Initialize an empty CKDS (creates the header and system keys)

 2 NOCVKEYS - Create NOCV-Enablement keys (for keys without CVs)
 3 ANSI     - Create ANSI system keys (for ANSI X9.17 services)
 4 ESYS     - Create enhanced system keys (for Symmetric services)

 5 REFRESH  - Activate an updated CKDS

Enter the name of the CKDS below.

CKDS ==> 'FIRST.EMPTY.CKDS'

```

Figure 50. ICSF Initialize a CKDS Panel

4. 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 should be the same name that is specified in the CKDSN installation option in the installation options data set. 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*.

5. Choose option 1, Initialize an empty CKDS, and press ENTER.

ICSF creates the header record in the disk copy of the CKDS. Next, ICSF sets the DES master key. ICSF then adds the required system keys 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 the master key has been set and before the system keys have been created, you will need to reset the master keys.

6. If you want ICSF to create NOCV-enablement keys when the initialization process has been completed, select option 2, NOCVKEYS, and press ENTER. The creation of NOCV-enablement keys is optional. It allows you to use either the key generator utility program or the Key Token Build callable service to create NOCV keys. NOCV keys allow you to send and receive keys from systems that do not use control vectors. For a description of NOCV keys, see the description of the NOCV keyword for the key generator utility program on 201.

Note: If you want to run the ICSF conversion program to convert a PCF CKDS into ICSF format, the CKDS you start ICSF with must contain NOCV-enablement keys. For more information about the conversion program, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

7. To create ANSI system keys that are used for the ANSI X9.17 services, choose option 3, ANSI.

The creation of ANSI system keys is optional. ANSI system keys are required if you intend to also create enhanced system keys.

The message ANSI KEYS ADDED appears on the top right of the panel, if the process succeeds.

8. To create enhanced system keys, choose option 4, ESYS.

The creation of enhanced system keys is optional. To create enhanced system keys, you must have previously installed the ANSI system keys in the CKDS.

The message ESYS KEYS ADDED appears on the top right of the panel, if the process succeeds.

When you complete the entire process, a master key and CKDS exist on your system. You can now generate keys using the key generate callable service and key generator utility program, or convert PCF keys to ICSF keys using the conversion program. ICSF services use the keys to perform the cryptographic functions you request.

Note: You enable special secure mode to initialize ICSF for the first time. When you perform the initialization process, you may choose to disable special secure mode.

PKDS

You normally have to initialize a PKDS only the first time you start ICSF on a system. However, depending on your system configuration, on a legacy machine that has a PKDS that doesn't have any keys, the PKDS will need to be initialized. Until this is done, PKA Callable Services cannot be enabled.

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.

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 142. For a description of how to use the TKE workstation to enter the master key, refer to *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

Steps for initializing the PKDS

To initialize the PKDS:

1. Return to the Primary Menu panel by pressing END from the Master Key Entry panel.
2. Select Option 2, MASTER KEY, on the Primary Menu panel as shown in Figure 51.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 2

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 51. Selecting the Master Key option on the primary menu panel

The Master Key Management panel appears. See Figure 52.

```
CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 1

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                   - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS          - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK                - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS          - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS          - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS             - Activate an updated PKA Cryptographic Key Data Set
```

Figure 52. ICSF Master Key Management Panel

3. Select option 5, INITIALIZE PKDS and the Initialize a PKDS panel appears. See Figure 53 on page 115.

```

CSFCMK30 ----- ICSF - Initialize a PKDS -----
COMMAND ===>

Enter the name of the PKDS to be initialized below.

PKDS ===> 'FIRST.EMPTY.PKDS'

```

Figure 53. ICSF Initialize 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.
5. The PKDS must now be refreshed to become active. Return to the previous panel and select option 7.

```

CSFCMK21 ----- ICSF - Refresh PKA Cryptographic Key Data Set -----
COMMAND ===>

Enter the name of the new PKDS below.

New PKDS ===> 'PKDS.NEW.MASTER'

Press ENTER to refresh the PKDS.
Press END to exit to the previous menu

```

Figure 54. Refresh PKDS

When you press ENTER, the PKDS is refreshed and becomes the in-storage copy.

6. In the New PKDS field, enter the name the initialized PKDS to make it the active PKDS.

Refreshing the CKDS at any time

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 10, “Managing Cryptographic Keys by Using the Key Generator Utility Program,” on page 191. For information on using the dynamic CKDS callable services, refer to the *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Steps for refreshing the CKDS

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: When you refresh a CKDS, consider temporarily disallowing dynamic CKDS update services. For more information, refer to “Steps for disallowing dynamic CKDS updates during KGUP updates” on page 192.

1. Enter option 2, MASTER KEY, on the ICSF Primary Menu panel to access the Master Key Management Panel.
2. Enter option 1, INIT/REFRESH CKDS to access the Initialize a CKDS panel, which is shown in Figure 55 on page 116.

```

CSFCKD00 ----- ICSF - Initialize a CKDS -----
COMMAND ==> 5

Enter the number of the desired option.

1 Initialize an empty CKDS (creates the header and system keys)

2 NOCVKEYS - Create NOCV-Enablement keys (for keys without CVs)
3 ANSI      - Create ANSI system keys (for ANSI X9.17 services)
4 ESYS      - Create enhanced system keys (for Symmetric services)

5 REFRESH   - Activate an updated CKDS

Enter the name of the CKDS below.

CKDS ==> 'PIN1.CKDS'

```

Figure 55. Selecting the Refresh Option on the ICSF Initialize a CKDS Panel

3. In the CKDS field, specify the name of the disk copy of the CKDS that you want ICSF to read into storage.
4. Choose option 5, REFRESH, and press ENTER.
ICSF places the disk copy of the specified CKDS into storage. During a REFRESH, ICSF does not load into storage any partial keys that may exist when you enter keys manually. 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.
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.
5. 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 refresh the CKDS. For information about these other methods, see “Refreshing the In-Storage CKDS” on page 225.

Refreshing the PKDS at any time

When you initialize a PKDS for the first time, you can copy the disk copy of the PKDS 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*. 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.

Note: Prior to refreshing a PKDS, consider temporarily disallowing PKDS write, create and delete services using the ICSF Administrative Control Functions panel.

1. Enter option 2, MASTER KEY MGMT, on the ICSF Primary Menu panel to access the Master Key Management Panel.

2. Enter option 7, REFRESH PKDS to access the Refresh PKA Cryptographic Key Data Set panel, which is shown in Figure 54 on page 115
3. In the New 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 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.
4. Press END to return to the Primary Menu panel.

Reentering master keys when they have been cleared

In these situations, the Cryptographic Coprocessor Feature clears the master key registers so that the master key values are not disclosed.

- If the Cryptographic Coprocessor Feature detects tampering
- If you issue a command from the TKE workstation to zeroize a domain
- If you issue a command from the Support Element to zeroize all domains

In these situations, the PCI Cryptographic Coprocessor Feature (PCICC) clears the master key registers so that the master key values are not disclosed.

- If the PCI Cryptographic Coprocessor 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 PCI Cryptographic Coprocessor Feature 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
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 keys in the CKDS are still enciphered under the cleared DES master key. The RSA and DSS private keys are also each enciphered under one of the cleared PKA master keys. Therefore, to recover the keys in the CKDS, and the PKA private keys, you must reenter the same master keys and set the DES master key. For security reasons, you may then want to change all the master keys.

PR/SM Considerations: If you are running in PR/SM logical partition (LPAR) mode, there are several situations (listed previously) that can cause loss of master keys and other data. In these cases, you must first ensure that key entry is enabled for each LP on the Change LPAR Crypto page on the support element Hardware Master Console. You must then reenter the master keys in each LP. 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 371.

Steps to reenter cleared master keys

Note: If PPINIT was used initially, you must rerun the utility with the same pass phrase to reenter the cleared master keys.

When the Cryptographic Coprocessor Feature clears the master keys, reenter the same master keys using these steps:

1. Check the status of the PKA callable services. If they are enabled, use the Administrative Control Functions to disable them. See “Steps for enabling and disabling PKA services” on page 124 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 101.
4. When you enter the new DES master key, select option 2, MASTER KEY, from the primary menu. The Master Key Management panel appears. See Figure 56. To activate the DES master key you just entered, you need to set it.
5. To set the DES master key, choose option 2 on the panel and press ENTER.

```
CSFMKM00 ----- ICSF - Master Key Management -----  
OPTION ==> 1  
  
Enter the number of the desired option above.  
  
 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or  
                               activate an updated Cryptographic Key Data Set  
 2 SET MK                     - Set a symmetric (DES or AES) master key  
 3 REENCIPHER CKDS           - Reencipher the CKDS prior to changing a symmetric  
                               master key  
 4 CHANGE MK                 - Change a symmetric master key and  
                               activate the reenciphered CKDS  
 5 INITIALIZE PKDS           - Initialize or update a PKA Cryptographic  
                               Key Data Set header record  
 6 REENCIPHER PKDS           - Reencipher the PKA Cryptographic Key Data Set  
 7 REFRESH PKDS              - Activate an updated PKA Cryptographic Key Data Set
```

Figure 56. Selecting the Set Host Master Key Option on the ICSF Master Key Management Panel

When you select option 2, ICSF checks that the states of the registers are correct. ICSF then transfers the DES master key from the new master key register to the master key register. This process sets the DES master key.

When ICSF attempts to set the DES master key, it displays a message on the top right of the Master Key Management panel. The message indicates either that the master key was successfully set, or that an error prevented the completion of the set process.

Notes:

- a. If your system is using both crypto modules provided by a Cryptographic Coprocessor Feature, ICSF sets the DES master key for each crypto module whose new DES master key enciphers the in-storage CKDS. You should reenter the DES master key into the new master key register for each of the crypto modules.
- b. The operator console receives messages that state that the crypto module is offline and then online for each crypto module. These actions should not affect cryptographic operations. However, if a crypto module does not have

either a current DES master key or a new DES master key that enciphers the current in-storage CKDS, the crypto module is left offline.

When you set the reentered DES master key, the DES master key that enciphers the existing CKDS now exists.

6. You can now change the DES master key, if you choose to, for security reasons. Continue with “Steps for changing master keys.”

Steps for changing master keys

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 when you reenter the cleared master keys.

There are three main steps involved in changing the DES master key:

1. Enter the DES and SYM-MK master key parts.
2. Reencipher the CKDS under the new DES master key.
3. Change the new DES master key and activate the reenciphered CKDS.

Note: When changing the master key, remember to change the name of the CKDS in the Installation Options Data Set.

There are six main steps involved in changing the PKA master keys:

1. Disable PKA Services
2. Enter the PKA master keys (SMK and KMMK, if equal to the SMK) and ASYM-MK.
3. Reencipher the PKDS under the new PKA master keys.
4. Refresh the PKDS.
5. Enable PKA Services
6. Enable PKDS read and write access.

Notes:

1. PKA master keys should only be changed if there is a PCICC available on the system.
2. When changing the master key, remember to change the name of the PKDS in the Installation Options Data Set.

DES master keys and the CKDS

The step-by-step procedure for changing the DES master key, reenciphering the CKDS, and activating the new DES master key are presented in “Steps for changing the DES master key and reenciphering the CKDS” on page 121. This topic provides some background on the contents of the master key registers during the key change process, and some compatibility mode considerations.

A DES master key and a CKDS that contains keys that are enciphered under that DES master key already exist. When you replace this existing DES master key with the new DES master key, you must reencipher the CKDS under the new DES master key.

Note: When you reencipher a CKDS, consider temporarily disallowing dynamic CKDS update services. For more information, refer to “Steps for disallowing dynamic CKDS updates during KGUP updates” on page 192.

For the CCF, if you changed the DES master key previously, the previous DES master key was stored in the auxiliary (or new/old) master key register. The currently active DES master key exists in the master key register. When you enter

the key parts of a new DES master key, they displace the previous DES master key in the auxiliary master key register. Therefore, the previous DES master key is lost. This is not true for the PCICC, which has separate registers for the old, new and current master key.

If you are using the Cryptographic Coprocessor Feature (CCF), to make the new DES master key the current active DES master key, you have ICSF swap the contents of the master key register and the auxiliary master key register. If you also have the PCICC, ICSF will change the PCI SYM-MKs. In this way, the new DES master key you have just entered becomes the current DES master key, and the previous DES master key is stored in the auxiliary master key register.

When the new DES master key is placed into the master key register, you must reencrypt all disk copies of the CKDS under the new DES master key. Then you are ready to activate the master key. When you change the master key, you have ICSF replace the in-storage copy of the CKDS with the reencrypted disk copy. This also makes the new master key active on the system.

The procedures you use to activate the new master key depend on your system's compatibility mode. 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 reencrypt 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 reencrypt internal key tokens under the new master key. ICSF applications that are holding internal key tokens which have been encrypted 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 are using the CCF, the safest method to use when changing the master key in either compatibility or coexistence mode is as follows:

1. Ensure that the name of the new CKDS is in the installation data set.
2. Re-IPL MVS.
3. Start CSF.

If you also have PCICC installed, when you start CSF, you must go to the Master Key Management panel (Figure 56 on page 118) and do a set (option 2). This will change the master keys of all the PCICC that match the CCF.

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 CSF again, applications using an operational key can either re-create or reimport the key.

Steps for changing the DES master key and reenciphering the CKDS

For information about reenciphering a CKDS in a sysplex environment, see Chapter 9, “Running in a Sysplex Environment,” on page 181.

1. Enter the key parts of the new master key that you want to replace the current master key. For information about how to do this procedure, see “Entering master key parts” on page 93.

The new master key register must be full when you change the master key.

2. Select option 3, REENCIPHER CKDS, on the Master Key Management panel, as shown in Figure 57, 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, when you reencipher a CKDS under a new master key, the new master key registers in all coprocessors must contain the same value.

```
CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 1

Enter the number of the desired option above.

1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
2 SET MK                    - Set a symmetric (DES or AES) master key
3 REENCIPHER CKDS           - Reencipher the CKDS prior to changing a symmetric
                             master key
4 CHANGE MK                 - Change a symmetric master key and
                             activate the reenciphered CKDS
5 INITIALIZE PKDS           - Initialize or update a PKA Cryptographic
                             Key Data Set header record
6 REENCIPHER PKDS           - Reencipher the PKA Cryptographic Key Data Set
7 REFRESH PKDS              - Activate an updated PKA Cryptographic Key Data Set
```

Figure 57. Selecting the Reencipher CKDS option on the ICSF Master Key Management Panel

3. The Reencipher CKDS panel appears. See Figure 58 on page 122.

```
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 ==> 'CKDS.CURRENT.MASTER'

Output CKDS ==> 'CKDS.NEW.MASTER'

Figure 58. Reencipher CKDS

4. 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.

Note: The output data set should already exist although it must be empty. For more information about defining a CKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

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.

6. If you have more than one CKDS on 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 Master Key Management panel.

Changing the master key 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 compatibility or co-existence mode, *do not* select option 4, the Change option. To activate the changed master key when running in compatibility or co-existence mode, you need to re-IPL MVS and start ICSF. When you re-IPL MVS and start ICSF, you activate the changed master key and refresh the in-storage CKDS. To do this, you must exit the panels at this time.

9. If you are running in noncompatibility mode, to change the master key select option 4, CHANGE MK, on the Master Key Management panel.

When you press the ENTER key, the Change Master Key panel appears. See Figure 59.

```
CSFCMK20 ----- ICSF Change Master Key -----  
COMMAND ==>
```

Enter the name of the new CKDS below:

```
New CKDS ==> 'CKDS.NEW.MASTER'
```

When the master key is changed, the new CKDS will become active.

Figure 59. Change Master Key Panel

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 Reencipher CKDS panel, which is shown in Figure 58 on page 122, 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.

Note: Each Cryptographic Coprocessor Feature includes two crypto modules, which ICSF recognizes as C0 and C1. You must enter the new master key into each of the coprocessors, when you perform the change. ICSF activates the new master key of both coprocessors that contain a new master key value that will encipher the CKDS. If you also have PCICCs on your system, load the new master key into all of the coprocessors.

If only one coprocessor new master key value matches the new CKDS, then that coprocessor will be used. The other coprocessor will remain offline until the new master key is changed to match the other coprocessor.

When the change occurs, the operator console receives messages that state that the Cryptographic Coprocessor Feature is offline and then online for each coprocessor. These actions should not affect cryptographic operations.

12. When changing the master key, remember to change the name of the CKDS in the Installation Options Data Set.

You can use a utility program to reencipher the CKDSs and change the master key instead of using the panels. “Reenciphering a disk copy of a CKDS and changing the master key” on page 339 describes how to use the utility program for these procedures.

PKA master keys and the PKDS

The step-by-step procedure for changing the PKA master keys is documented in this topic. The procedure assumes that SMK=KMMK. It is recommended that the KMMK=SMK to maximize the routing capability to the PCICC and to enable PKDS reencipher. Once that is completed, it is necessary to reencipher and activate the PKDS.

If the SMK does not equal KMMK, see “Steps for setting the SMK equal to the KMMK” on page 131.

Attention: If you do not have a PCICC, you should not change the PKA Master Keys. Changing the PKA master keys will make all internal tokens in the current PKDS unusable. You will need to reencipher and activate the PKDS in order to use them with the changed master key. This requires a PCICC on your system. See “Steps for reenciphering and refreshing the PKDS” on page 129 for more information.

When the PKDS is shared by multiple images in a sysplex environment, the PKA master key must also be changed on all the sharing systems. See Chapter 9, “Running in a Sysplex Environment,” on page 181.

Steps for enabling and disabling PKA services

When you enter or change the PKA master keys or the ASYM-MK, you must first disable the PKA services. To enable or disable PKA services:

1. Access the administrative control functions by choosing option 4, ADMINCNTL, on the Primary Menu panel, as shown in Figure 60.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 4

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 60. Selecting Administrative Control on the ICSF Primary Menu Panel

The Administrative Control Function panel appears. See Figure 61 on page 125.

```

CSFACF00 ----- ICSF Administrative Control Functions
COMMAND ==>
    Active CKDS: CRYPTO25.HCR7704.CKDS
    Active PKDS: CRYPTO25.HCR7704.PKDS
    Active TKDS: CRYPTO25.HCR7704.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 61. Enabling and Disabling the PKA Callable Services

2. Enter the appropriate character and press ENTER.
 - To enable the PKA callable services, enter an 'E' before the PKA Callable Services function.

Note: If using a PKDS, you must also enable Dynamic PKDS Access.

- To disable the PKA callable services, enter a 'D' before the PKA Callable Services function.

Note: Disabling PKA callable services also disables Dynamic PKDS Access.

Steps for changing PKA master keys

To change the PKA master keys:

1. Disable the PKA callable services as described previously.
2. Return to the primary menu and select option 1, COPROCESSOR MGMT, and press enter.

The Coprocessor Management panel appears.

```

CSFCMP00 ----- ICSF Coprocessor Management -----
COMMAND ==>

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                                STATUS
-----
_ A06                                                ACTIVE
_ A07                                                ACTIVE
E C0          E589C396944007A6 5D40369997A386F4      ACTIVE
E C1          0AA379BFD2387960 0367DC04533125FF      ACTIVE
E P00         41-00YE1                                ACTIVE
E P01         41-00K11                                ACTIVE
E P02         41-0A355                                ACTIVE
_ P03         41-0BA3F                                ONLINE
_ P04         41-0RT2T                                DEACTIVATED
_ P05         41-00342                                DISABLED

```

Figure 62. Selecting the coprocessor on the Coprocessor Management Panel

3. Select the coprocessor(s) for PKA master key entry by entering 'E' before the coprocessor and pressing enter.

The Master Key Entry panel appears. See Figure 63. You need to RESET to clear the contents of the registers so you can set a new key value.
 In this example, ALL-PKA has been entered, as SMK=KMMK. If this was not the case, SMK would have been used.

```

CSFDKE10 ----- ICSF - Master Key Entry -----
COMMAND ==>

                CCF DES/PCICC SYM-MK new master key register      : EMPTY
                CCF Signature/PCICC ASYM-MK master key register  : NOT THE SAME
                CCF Key management master key register            : FULL

Specify information below
Key Type ==> ALL-PKA      (DES, SMK, KMMK, ALL-PKA)

Part      ==> RESET      (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
           ==> 0000000000000000
           ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)
  
```

Figure 63. The Master Key Entry Panel to Reset Registers

4. When you select RESET, the Restart Key Entry Process panel is displayed. See Figure 64.
 This panel confirms your request to restart the key entry process. Press ENTER.

```

CSFDKE40 ----- ICSF - Restart Key Entry Process -----

ARE YOU SURE YOU WISH TO RESTART THE KEY ENTRY PROCESS?

Restarting the process will clear the ALL-PKA master key register.

WARNING: Resetting the KMMK or SMK will invalidate any private
         internal key tokens in the PKDS

Press ENTER to confirm restart request
Press END   to cancel restart request
  
```

Figure 64. Confirm Restart Request Panel

5. The Master Key Entry panel again appears. See Figure 65 on page 127. Enter the type of PKA master key you are changing and enter the key part.


```

CSFDKE10 ----- ICSF - Master Key Entry -----
COMMAND ==>

                CCF DES/PCICC SYM-MK new master key register      : EMPTY
                CCF Signature/PCICC ASYM-MK master key register   : EMPTY
                CCF Key management master key register            : EMPTY

Specify information below
Key Type ==> ALL-PKA      (DES, SMK, KMMK, ALL-PKA)

Part      ==> FIRST      (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 59

Key Value ==> 8F887096A8D4922B
           ==> 75D1189666F4DAA7
           ==> 9B28AEFA8C47760F (SMK, KMMK and ALL-PKA only)

```

Figure 65. The Master Key Entry Panel with First Key Values

6. Fill in the panel
 - a. Enter the master key type in the Key Type field.
In this example we are entering ALL-PKA. A PKA master key requires at least two key parts. You may enter additional key parts if necessary. ALL-PKA includes the SMK, KMMK and ASYM-MK.
 - b. Enter FIRST in the Part field.
 - c. Enter the two-digit checksum and the three 16-digit key values (if you did not use random number generate).
 - d. Make sure you have recorded the three 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 cryptographic coprocessor calculated, the key part is accepted. The message at the top of the panel will now state KEY PART LOADED.
The Signature/PCICC ASYM-MK register status and KMMK status change to PART FULL. The hash pattern that is calculated for the key part appears near the bottom of the panel. Compare it with the pattern generated by the checksum, VP, HP utility or provided by the person who gave you the key part value to enter.
 - f. Record the hash pattern.
7. 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.

8. Now enter the FINAL key part.

```
CSFDKE10 ----- ICSF - Master Key Entry -----  
COMMAND ==>  
  
          CCF DES/PCICC SYM-MK new master key register      : EMPTY  
          CCF Signature/PCICC ASYM-MK master key register   : NOT THE SAME  
          CCF Key management master key register           : FULL  
  
Specify information below  
Key Type ==> ALL-PKA      (DES, SMK, KMMK, ALL-PKA)  
  
Part      ==> FINAL      (RESET, FIRST, MIDDLE, FINAL)  
  
Checksum  ==> 53  
  
Key Value ==> 8FDAD096A8D4922B  
          ==> 75D1189ADAF4DAA7  
          ==> 9B28333A8C47760F (SMK, KMMK and ALL-PKA only)
```

Figure 66. The Master Key Entry Panel with Final Key Values

9. Fill in the panel

- a. Enter the master key type in the Key Type field.
In this example we are entering ALL-PKA. ALL-PKA includes the SMK, KMMK and ASYM-MK.
- b. Enter FINAL in the Part field.
- c. Enter the two-digit checksum and the three 16-digit key values (if you did not use random number generate).
- d. Make sure you have recorded the three 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 cryptographic coprocessor calculated, the key part is accepted. The message at the top of the panel states KEY PART LOADED, as shown in Figure 66.

The Signature/PCICC ASYM-MK master key register status changes to NOT THE SAME. This is because the PCICC current ASYM-MK register is loaded with the value in the new master key register and the new ASYM-MK register is empty. The KMMK status changes to FULL.

The hash pattern that is calculated for the key part appears near the bottom of the panel. Compare it with the pattern generated by the checksum, VP, HP utility or provided by the person who gave you the key part value to enter.

- f. Record the hash pattern.

10. 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.
11. When you have entered the PKA master keys correctly, the PKA master key registers are active when the final key part is loaded. You must then reencipher and activate the PKDS (“Steps for reenciphering and refreshing the PKDS”) and enable PKA callable services “Steps for enabling and disabling PKA services” on page 124. Also enable PKDS Read and PKDS Write, Create and Delete.
 12. When changing the master key, remember to change the name of the PKDS in the Installation Options Data Set.

Steps for reenciphering and refreshing the PKDS

When changing the PKA master keys, you must reencipher the private keys.

Note: Beginning with HCR7750, LRECL length in the PKDS has increased. You can share the larger PKDS with down-level systems by installing the toleration APAR OA21807. Even with toleration APAR OA21807 installed, however, be aware that reencipherment of a larger PKDS must always be performed on an HCR7750 or later system.

1. To reencipher the PKDS when the PKA SMK and ASYM-MK have been changed, go to the Master Key Management panel and select option 6.

Note: Only keys enciphered under the SMK and the ASYM-MK are reenciphered. PKDS reencipher will not be able to reencipher private keys encrypted under the CCF key management key (KMMK) if the KMMK does not equal the SMK. If this is the case, see “Steps for setting the SMK equal to the KMMK” on page 131 when you reencipher.

```
CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 6

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                    - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS          - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK                - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS          - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS          - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS             - Activate an updated PKA Cryptographic Key Data Set
```

Figure 67. Selecting the Reencipher PKDS Option on the Master Key Management Panel

2. The Reencipher PKDS panel appears. In the Input PKDS field, specify the name of the PKDS that you want ICSF to reencipher under the current SMK and ASYM-MK.

In the Output PKDS field, specify the name of an empty VSAM data set. ICSF places the reenciphered keys in this data set.

```
CSFCMK11 ----- ICSF - Reencipher PKDS -----
COMMAND ==>

To reencipher all PKDS entries from encryption under the old signature/
asymmetric-keys master key to encryption under the current master key, enter
the PKDS names below.

Input PKDS   ==> 'PKDS.CURRENT.MASTER'

Output PKDS ==> 'PKDS.NEW.MASTER'

Press ENTER to reencipher the PKDS.
Press END   to exit to the previous menu
```

Figure 68. Reencipher PKDS

Press enter to reencipher the PKDS. Once successful, you have to refresh the PKDS. Return to the Master Key Management panel and select option 7.

```
CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 7

Enter the number of the desired option above.

1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                           activate an updated Cryptographic Key Data Set
2 SET MK                  - Set a symmetric (DES or AES) master key
3 REENCIPHER CKDS         - Reencipher the CKDS prior to changing a symmetric
                           master key
4 CHANGE MK               - Change a symmetric master key and
                           activate the reenciphered CKDS
5 INITIALIZE PKDS         - Initialize or update a PKA Cryptographic
                           Key Data Set header record
6 REENCIPHER PKDS         - Reencipher the PKA Cryptographic Key Data Set
7 REFRESH PKDS           - Activate an updated PKA Cryptographic Key Data Set
```

Figure 69. Selecting the Activate PKDS Option on the Master Key Management Panel

The Refresh PKDS panel appears. Enter the name of the PKDS that you want ICSF to use. The PKDS must have already been reenciphered under the current Signature/Asymmetric-keys master key.

```
CSFCMK21 ----- ICSF - Refresh PKA Cryptographic Key Data Set -----
COMMAND ==>

Enter the name of the new PKDS below.

New PKDS ==> 'PKDS.NEW.MASTER'

Press ENTER to refresh the PKDS.
Press END   to exit to the previous menu
```

Figure 70. Refresh PKDS

When you press ENTER, the PKDS becomes active.

Steps for setting the SMK equal to the KMMK

It is highly recommended that the KMMK, SMK and ASYM-MK be equal. This will facilitate migration to new features on crypto hardware.

If you are a new user and using Pass Phrase Initialization, ensure that you answer Y for Signature MK = Key Management MK? on Figure 12 on page 75. If using Clear Key Entry, make sure that you enter the same value for your SMK and KMMK.

If you are an existing user and for some reason your KMMK does not equal the SMK and ASYM-MK, you should follow this procedure. You must have a PCICC on your system.

1. Disable PKA services (see “Steps for enabling and disabling PKA services” on page 124).
2. Determine the value of the SMK
 - a. If you used Pass Phrase Initialization, go to the main menu and choose option 5, UTILITY. Select option 5, PPKEYS.

```
CSFUTL00 ----- ICSF - Utilities -----
OPTION ==> 5

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 71. ICSF Utilities Panel

The Master Key Values from Pass Phrase panel appears (Figure 72).

```
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
```

Figure 72. ICSF Master Key Values from Pass Phrase Panel

Enter the previously used pass phrase and your SMK and KMMK values will be displayed.

- b. If you used Master Key entry, you must retrieve the value from your written files.

3. Use the value of the SMK as the new KMMK and ASYM-MK values (see “PKA master keys and the PKDS” on page 124).
4. Reencipher and Activate the PKDS (see “Steps for reenciphering and refreshing the PKDS” on page 129).

Steps for clearing master keys

For security reasons, your installation may need to clear the master keys. This may be required, for example, when 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 PCIX 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 a new DES master key, reencipher a dummy CKDS, and change the master key. You would need to perform this operation twice to ensure that the master key is cleared from the auxiliary (old) master key register. You would also need to reset both of the PKA master keys and process the PCICC master keys.

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.

Steps for adding a PCICC after CCF initialization

You may need to initialize PCI Cryptographic Coprocessors after system initialization.

Note: Use this procedure if you did not run the Pass Phrase Initialization utility. If you used the utility, see Chapter 5, “Using the Pass Phrase Initialization Utility,” on page 73.

Follow this procedure.

1. Select option 1, COPROCESSOR MGMT, on the Primary Menu panel.
2. The Coprocessor Management panel, as shown in Figure 73 on page 133, appears.

```
CSFCMP00 ----- ICSF Coprocessor Management -----
COMMAND ==>

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                      STATUS
-----
_ A06                                               ACTIVE
_ A07                                               ACTIVE
_ C0          E589C396944007A6 5D40369997A386F4      ACTIVE
_ C1          0AA379BFD2387960 0367DC04533125FF      ACTIVE
_ P00         41-00YE1                                ACTIVE
_ P01         41-00K11                                ACTIVE
_ P02         41-0A355                                ACTIVE
E P03         41-0BA3F                                ONLINE
_ P04         41-0RT2T                                DEACTIVATED
_ P05         41-00342                                DISABLED
```

Figure 73. Selecting a coprocessor on the Coprocessor Management Panel

- 3. Select the Coprocessor to be processed by entering 'E' next to the Coprocessor.
- 4. The Master Key Entry panel appears. See Figure 74.

```
CSFDKE10----- ICSF - Master Key Entry -----
COMMAND ==>

CCF DES/PCICC SYM-MK new master key register      : EMPTY
CCF Signature/PCICC ASYM-MK master key register   : EMPTY
CCF Key management master key register            : EMPTY

Specify information below
Key Type ==> ____ (DES, SMK, KMMK, ALL-PKA)

Part      ==> _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
           ==> 0000000000000000
           ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)

Press ENTER to process.
Press END   to exit to the previous menu.
```

Figure 74. The Master Key Entry Panel to Reset Registers

- Ensure that the CCF Signature/PCICC ASYM-MK master key register field indicates EMPTY. If it does not, you will need to RESET to clear the contents of the registers to set a new key value.
- 5. You must now load the SYM-MK and ASYM-MK keys into your system.
If you are going to reload your current master keys, you need to know the current master key value and checksum. If you want the PCICC to become ACTIVE after CCF initialization, you MUST enter the same master key values.

Follow the instructions on “Steps for entering the first master key part” on page 101.

6. When all key parts have been loaded, SET the master key. From the Primary Menu panel choose option 2 - Master Key. From the Master Key Management panel, choose option 2 - SET MK.

Chapter 7. Managing Master Keys - PCIXCC, CEX2C, or CEX3C

This topic describes how to use the Master Key Entry panels to enter master keys in a PCIXCC, CEX2C, or CEX3C. Each PCIXCC, CEX2C, or CEX3C is capable of performing cryptographic functions and holding master keys within a secure boundary.

You can have multiple PCIXCCs/CEX2Cs on the z990 and z890. Multiple CEX2Cs are available on z9 EC, z9 BC, z10 EC and z10 BC systems. Multiple CEX3Cs are available on z10 EC or z10 BC systems. Requests for cryptographic services are routed to the PCIXCC, CEX2C, or CEX3C.

Note: The PCI Cryptographic Accelerators improve private key decryption performance. They do not require setting of master keys.

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 how you get the master key parts, **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.

When you enter the asymmetric-keys master key (ASYM-MK) the first time, the PKA callable services are initially disabled. Once you have entered the ASYM-MK, you must enable the PKA callable services for these services to work. When you change the ASYM-MK, you need to disable the PKA callable services. To enable and disable the PKA callable services refer to “Steps for enabling and disabling PKA services” on page 168.

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 142.

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 bytes long. ICSF defines these master keys by exclusive ORing two or more key parts. Each of the master key parts is also 16 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. We refer to even or mixed parity keys as non-odd parity keys.

Attention: The PCIXCC, CEX2C, or CEX3C will not allow certain 'weak' keys as DES and asymmetric master keys. The list of weak keys are documented in Appendix F, "Questionable (Weak) Keys," on page 381.

The AES master key (AES-MK) is 32 bytes long. ICSF defines these master keys by exclusive ORing two or more key parts.

The ASYM-MK is 24 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 ASYM-MK, the coprocessor calculates a sixteen-byte hash pattern.

Before the verification and hash patterns can be calculated, the DES-MK master key must have been set.

The ICSF Master Key Entry panel displays the verification pattern or hash pattern. Check the displayed verification pattern against the optional verification pattern you may have generated at the time you generated the DES-MK or AES-MK master key parts and the checksum. Check the displayed hash pattern against the optional hash pattern that you may have generated at the same time you generated the ASYM-MK and the checksum. The verification pattern or hash 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.

Note: Some keys stored in the CKDS are enciphered under the DES-MK or AES-MK master key. The master key verification patterns are stored in the CKDS header record. Checking the verification pattern is optional; it is not required for key entry.

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*.

Note: ICSF must be initialized with a DES-MK or AES-MK master key to use the random number generate callable service or the Random Number Generator panel.

These topics describe using the ICSF utilities to generate key parts, checksums, verification patterns, and hash patterns.

Steps for generating key parts using ICSF utilities

1. Access ICSF utilities by choosing option 5, UTILITY, on the Primary Menu panel, as shown in Figure 75.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 5

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 75. Selecting the Utility Option on the ICSF Primary Menu Panel

The Utilities panel appears. See Figure 76 on page 138. You use the RANDOM and CHECKSUM options to generate random numbers, checksums, and verification patterns for master key management.

```

CSFUTL00 ----- ICSF - Utilities -----
OPTION ==> 3

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 76. ICSF Utilities Panel

- Choose option 3, RANDOM, to access the Random Number Generator panel, shown in Figure 77.

```

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

```

Figure 77. ICSF Random Number Generator Panel

- To select the parity of the random numbers, enter ODD, EVEN, or RANDOM next to Parity Option and press ENTER.
 The DES-MK 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 or PKA 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. The PKA master key uses random numbers 1 through 3. The AES master key uses 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 78. 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. The random number generator and the hash and verification pattern, however, do not work until ICSF has been initialized with a valid master key.

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 365 for a description of the checksum algorithm. See “Algorithm for calculating a verification pattern” on page 367 for a description of the algorithm for the verification pattern. See “The MDC-4 Algorithm for Generating Hash Patterns” on page 368 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 79.

```
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 79. Selecting the Checksum Option on the ICSF Utilities Panel

The Checksum and Verification and Hash Pattern panel appears. See Figure 80 on page 140.

```

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 and PKA keys only)
               ==> 8453313235ABF69C  Input key value 4 (AES keys only)

Checksum      : 00                                Check digit for key value
Key Part VP   : 0000000000000000  Verification Pattern
Key Part HP   : 0000000000000000  Hash Pattern
               : 0000000000000000

```

Figure 80. 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:
 - MASTER to generate a checksum and hash and verification pattern for a DES master key part.
 - AES-MK to generate a checksum and verification pattern for an AES master key part
 - PKAMSTR to generate a checksum and hash pattern for an ASYM-MK key part.

If you leave the Key Type field blank and press ENTER, the Key Type Selection panel appears. See Figure 81 on page 141.

```

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
  EXPORTER     Export key encrypting key
  IMP-PKA      Limited authority importer key
  IMPORTER     Import key encrypting key
  IPINENC      Input PIN encrypting key
s MASTER      DES Master key
  OPINENC      Output PIN encrypting key
  PINGEN       PIN generation key
  PINVER       PIN verification key
  PKAMSTR      PKA master key
***** BOTTOM OF DATA *****

```

Figure 81. Key Type Selection Panel Displayed During Hardware Key Entry

4. Type 'S' to the left of the MASTER key type, and press ENTER to return to the Checksum and Verification Pattern panel as shown in Figure 82.
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      ==> MASTER      (Selection panel displayed if blank)

Key Value      ==> 51ED9CFA90716CFB  Input key value 1
                ==> 58403BFA02BD13E8  Input key value 2
                ==> 0000000000000000  Input key value 3 (AES and PKA keys only)
                ==> 0000000000000000  Input key value 4 (AES keys only)

Checksum       : 40                      Check digit for key part
Key Part VP    : 0CCE190A635A6C89      Verification Pattern
Key Part HP    : EA58E51179754FB7      Hash Pattern
                : C102957465CE479E

```

Figure 82. 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 of a tamper. If the PCIXCC, CEX2C, or CEX3C 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” on page 142.
- If you have generated an intermediate master key part, continue with “Steps for entering intermediate key parts” on page 145.

- If you have generated a final master key part, continue with “Steps for entering the final key part” on page 147.

Steps for entering the first master key part

Use the Master Key Entry panels to enter each key part.

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, as shown in Figure 83, and press ENTER.

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

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 83. Selecting the Coprocessor Management option on the primary menu panel

The ICSF Coprocessor Management panel appears (Figure 84 on page 143).

2. Select the coprocessor(s) 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 ONLINE. When the DES or AES Master Key is set, the status will be ACTIVE.


```

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

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

COPROCESSOR  SERIAL NUMBER                                STATUS
-----
- A06                                               ACTIVE
- A07                                               ACTIVE
E X02          42-K0001                               ONLINE
- X04          42-K0043                               DEACTIVATED
- E05          42-K0058                               DISABLED

```

Figure 84. Selecting the coprocessor on the Coprocessor Management Panel

3. The ICSF Master Key Entry panel appears. See Figure 85.

```

CSFDKE50----- ICSF - Master Key Entry -----
COMMAND ==>

AES new master key register      : EMPTY
DES new master key register      : EMPTY
Asymmetric new master key register : EMPTY

Specify information below
Key Type ==> ____ (AES-MK, ASYM-MK, DES-MK)

Part ==> ____ (RESET, FIRST, MIDDLE, FINAL)

Checksum ==> 40

Key Value ==> 51ED9CFA90716CFB
           ==> 58403BFA02BD13E8
           ==> 0000000000000000 (AES-MK and ASYM-MK only)
           ==> 0000000000000000 (AES-MK only)

Press ENTER to process.
Press END to exit to the previous menu.

```

Figure 85. Master Key Entry Panel

If you are not running on z10 EC or z10 BC with the Nov. 2008 or later licensed internal code (LIC), the panel will have a slightly different appearance:

```

CSFDKE50----- ICSF - Master Key Entry -----
COMMAND ==>

                AES new master key register           : NOT SUPPORTED
                DES new master key register           : EMPTY
                Asymmetric new master key register     : EMPTY

Specify information below
Key Type ==> ____ (AES-MK, ASYM-MK, DES-MK)

Part      ==> _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum ==> 40

Key Value ==> 51ED9CFA90716CFB
           ==> 58403BFA02BD13E8
           ==> 0000000000000000 (AES-MK and ASYM-MK only)
           ==> 0000000000000000 (AES-MK only)

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 86. Master Key Entry Panel

Error messages will be issued if you try to create or change AES master keys.

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 random number generate).
- 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 PCIXCC, CEX2C, or CEX3C calculated, the key part is accepted. The message at the top of the panel states KEY PART LOADED, as shown in Figure 87 on page 145. 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
      Asymmetric new master key register : EMPTY

Specify information below
Key Type ==> DES-MK      (AES-MK, ASYM-MK, DES-MK)

Part      ==> FIRST      (RESET, FIRST, MIDDLE, FINAL)

Checksum ==> 00

Key Value ==> 0000000000000000
           ==> 0000000000000000
           ==> 0000000000000000 (AES-MK and ASYM-MK only)
           ==> 0000000000000000 (AES-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 87. 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 137 if you are using the ICSF utilities to generate random numbers for key values.
- “Steps for entering intermediate key parts” 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 when 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(s) 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 88).

```

CSFDKE50 ----- ICSF - Master Key Entry -----
COMMAND ==>

                AES new master key register           : EMPTY
                DES new master key register           : PART FULL
                Asymmetric new master key register     : EMPTY

Specify information below
Key Type ==> ____ (AES-MK, ASYM-MK, DES-MK)

Part      ==> ____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 58

Key Value ==> 12021945CADE8431
           ==> 04091939BABE9632
           ==> 0000000000000000 (AES-MK and ASYM-MK only)
           ==> 0000000000000000 (AES-MK only)

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 88. 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-MK 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 generate).
 - 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 PCIXCC, CEX2C, or CEX3C calculated, the key part is accepted. The message at the top of the panel states KEY PART LOADED, as shown in Figure 89 on page 147. 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
                Asymmetric new master key register     : EMPTY

Specify information below
Key Type ==> ____ (AES-MK, ASYM-MK, DES-MK)

Part      ==> _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum ==> 58

Key Value ==> 0000000000000000
           ==> 0000000000000000
           ==> 0000000000000000 (AES-MK and ASYM-MK only)
           ==> 0000000000000000 (AES-MK only)

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 89. 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 137 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

When 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(s) 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
                Asymmetric new master key register     : EMPTY

Specify information below
Key Type  ===>  ____ (AES-MK, ASYM-MK, DES-MK)

Part      ===>  _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ===>  65

Key Value ===>  1939040919720419
              ===>  EA10111975BB5312
              ===>  0000000000000000 (AES-MK and ASYM-MK only)
              ===>  0000000000000000 (AES-MK only)

                Press ENTER to process.
Press END    to exit to the previous menu.

```

Figure 90. 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-MK 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 generate).
- 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 PCIXCC, CEX2C, or CEX3C calculated, the key part is accepted. The message at the top of the panel states KEY PART LOADED, as shown in Figure 91 on page 149. 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
          Asymmetric new master key register     : EMPTY

Specify information below
Key Type ==> DES-MK          (AES-MK, ASYM-MK, DES-MK)

Part      ==> FINAL          (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
          ==> 0000000000000000
          ==> 0000000000000000 (AES-MK and ASYM-MK only)
          ==> 0000000000000000 (AES-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 91. 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, since 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 150.
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, ICSF places the verification pattern for the DES-MK and AES-MK master key into the CKDS 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-MK or AES-MK for the first time, initialize the CKDS with the DES-MK master key. For a description of the process of initializing a DES-MK or AES-MK master key on your system, see “Initializing the CKDS and PKDS at First-Time Startup” on page 152.
- If you are defining a DES-MK or AES-MK master key when it was cleared, set the DES-MK or AES-MK master key to make it active. For a description of the process of recovering from tampering, see “Reentering master keys when they have been cleared” on page 162.
- If you are changing a DES-MK master key, reencipher the CKDS under the new DES-MK or AES-MK master key and make it active. For a description of the process of changing a DES-MK or AES-MK master key, see “Steps for changing master keys” on page 163.

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-MK 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
                Asymmetric new master key register     : EMPTY

Specify information below
Key Type ==> DES-MK          (AES-MK, ASYM-MK, DES-MK)

Part      ==> RESET_        (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 40

Key Value ==> 51ED9CFA90716CFB
           ==> 58403BFA02BD13E8
           ==> 0000000000000000 (AES-MK and ASYM-MK only)
           ==> 0000000000000000 (AES-MK only)

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 92. Selecting Reset on the Master Key Entry Panel

3. Press ENTER.

The Restart Key Entry Process panel appears. See Figure 93. 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 93. 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 94 on page 152. 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
                Asymmetric new master key register     : EMPTY

Specify information below
Key Type ==> _____ (AES-MK, ASYM-MK, DES-MK)

Part      ==> _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
            ==> 0000000000000000
            ==> 0000000000000000 (AES-MK and ASYM-MK only)
            ==> 0000000000000000 (AES-MK only)

```

Figure 94. 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.

Initializing the CKDS and PKDS at First-Time Startup

If running in a sysplex, see Chapter 9, “Running in a Sysplex Environment,” on page 181.

The first time you start ICSF, you must:

- Create a cryptographic key data set (CKDS)
- Create a PKA key data set (PKDS)
- Enter a new DES-MK into each PCIXCC, CEX2C, or CEX3C (optional)
- Enter a new AES-MK into each CEX2C, or CEX3C (optional)
- Initialize the CKDS
- Enter a new ASYM-MK into each PCIXCC, CEX2C, or CEX3C
- Initialize the PKDS

Note: Once these tasks are completed, you should enable PKA callable services and PKDS read and write access.

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.

If you are running on a IBM @server zSeries 990, IBM @server zSeries 890, z9 EC, z9 BC, z10 EC and z10 BC. and wish to share your CKDS and PKDS with a CCF system on an IBM @server zSeries 900 or z800, you should initialize the CKDS and PKDS on the IBM @server zSeries 900 or z800.

CKDS

You only have to initialize a CKDS the first time you start ICSF on a system. 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 disk copy into storage. For information about how to read a disk copy into storage, see “Refreshing the CKDS at Any Time” on page 160.

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 142. For a description of how to use the TKE workstation to enter the master key, refer to *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

Steps for initializing a CKDS

For information about initializing a CKDS in a sysplex environment, see Chapter 9, “Running in a Sysplex Environment,” on page 181.

To initialize the CKDS:

1. Return to the Primary Menu panel by pressing END from the Master Key Entry panel.
2. Select Option 2, MASTER KEY MGMT, on the Primary Menu panel as shown in Figure 95.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 2

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

      5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 95. Selecting the Master Key option on the primary menu panel

The Master Key Management panel appears. See Figure 96 on page 154.

```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 1

Enter the number of the desired option above.

1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
2 SET MK                    - Set a symmetric (DES or AES) master key
3 REENCIPHER CKDS          - Reencipher the CKDS prior to changing a symmetric
                             master key
4 CHANGE MK                - Change a symmetric master key and
                             activate the reenciphered CKDS
5 INITIALIZE PKDS          - Initialize or update a PKA Cryptographic
                             Key Data Set header record
6 REENCIPHER PKDS          - Reencipher the PKA Cryptographic Key Data Set
7 REFRESH PKDS             - Activate an updated PKA Cryptographic Key Data Set

```

Figure 96. ICSF Master Key Management Panel

3. Select option 1, INIT/REFRESH/UPDATE CKDS and the Initialize a CKDS panel appears. See Figure 97. If AES master keys are supported, a different panel appears (Figure 98).

```

CSFCKD10 ----- ICSF - Initialize a CKDS -----
COMMAND ==>

Enter the number of the desired option.

1 Initialize an empty CKDS (creates the header and system keys)
  Record authentication required (Y/N)
2 REFRESH - Activate an updated CKDS

Enter the name of the CKDS below.

CKDS ==> 'FIRST.EMPTY.CKDS'

```

Figure 97. ICSF Initialize a CKDS Panel

```

CSFCKD20 ----- ICSF - Initialize a CKDS -----
COMMAND ==>

Enter the number of the desired option.

1 Initialize an empty CKDS
  Record authentication required? (Y/N) ==>
2 REFRESH - Activate an updated CKDS
3 Update an existing CKDS

Enter the name of the CKDS below.

CKDS ==> 'FIRST.EMPTY.CKDS'

```

Figure 98. ICSF Initialize a CKDS Panel if AES master keys are supported

4. 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*.

5. Choose option 1, Initialize an empty CKDS, and press ENTER.

To improve performance, answer **N** to Record authentication required.

ICSF creates the header record in the disk copy of 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 conversion program. ICSF services use the keys to perform the cryptographic functions you request.

Updating the CKDS with the AES master key

On systems that support the AES master key, you can add the AES master key to any existing CKDS. It is also possible to add the DES master key to a CKDS that was initialized with only the AES master key.

These are the steps to update the CKDS:

1. Load the new AES master key by using the master key entry panels or by using TKE. The AES master key must be loaded on all active coprocessors.
2. From the Primary Menu, select option 2, MASTER KEY MGMT:

```

CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 2

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 99. Selecting the Master Key option on the primary menu panel

3. Select option 1, INIT/REFRESH/UPDATE CKDS.

```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 1

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                   - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS          - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK                - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS          - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS          - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS             - Activate an updated PKA Cryptographic Key Data Set

```

Figure 100. ICSF Master Key Management Panel

4. The Initialize a CKDS panel appears. In the CKDS field, enter the name of an existing, initialized CKDS.

```

CSFCKD20 ----- ICSF - Initialize a CKDS -----
COMMAND ==>

Enter the number of the desired option.

1 Initialize an empty CKDS
   Record authentication required? (Y/N) ==>
2 REFRESH - Activate an updated CKDS
3 Update an existing CKDS

Enter the name of the CKDS below.

CKDS ==> 'FIRST.EMPTY.CKDS'

```

Figure 101. ICSF Initialize a CKDS Panel if AES master keys are supported

5. Choose option 3, Update an existing CKDS 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 CKDS header record. Note that all the CKDS' that you wish to update should be processed prior to going to step 6.
6. In the CKDS field, enter the name of the updated CKDS that will be the active CKDS.
7. Select option 2, REFRESH and press **ENTER**. The in-storage copy of the CKDS will be updated with your updated CKDS.

```

CSFCKD20 ----- ICSF - Initialize a CKDS -----
COMMAND ==>

Enter the number of the desired option.

1 Initialize an empty CKDS
   Record authentication required? (Y/N) ==>
2 REFRESH - Activate an updated CKDS
3 Update an existing CKDS

Enter the name of the CKDS below.

CKDS ==> 'FIRST.EMPTY.CKDS'

```

Figure 102. ICSF Initialize a CKDS Panel

8. Return to the Master Key Management panel by pressing **END**. Choose option 2, SET MK and press **ENTER**. ICSF sets the AES master key and your system can be used to encrypt AES key operations.

```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 2

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                   - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS         - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK               - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS         - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS         - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS           - Activate an updated PKA Cryptographic Key Data Set

```

Figure 103. ICSF Master Key Management Panel

PKDS

You only have to initialize a PKDS the first time you start ICSF on a system.

Note: You must have a valid ASYM-MK loaded to initialize 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.

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 142. For a description of how to use the TKE workstation to enter the master key, refer to *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

Steps for initializing a PKDS

To initialize the PKDS:

1. Return to the Primary Menu panel by pressing END from the Master Key Entry panel.
2. Select Option 2, MASTER KEY MGMT, on the Primary Menu panel as shown in Figure 104 on page 159.


```

CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 2

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 104. Selecting the Master Key option on the primary menu panel

The Master Key Management panel appears. See Figure 105.

```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 5

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                   - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS          - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK                - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS          - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS          - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS             - Activate an updated PKA Cryptographic Key Data Set

```

Figure 105. ICSF Master Key Management Panel

3. Select option 5, INITIALIZE PKDS and the Initialize a PKDS panel appears. See Figure 106.

```

CSFCMK30 ----- ICSF - Initialize a PKDS -----
COMMAND ==>

Enter the name of the PKDS to be initialized below.

PKDS ==> 'FIRST.EMPTY.PKDS'

```

Figure 106. ICSF Initialize 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.
5. The PKDS must now be refreshed to become active. Return to the previous panel and select option 7.

```
CSFCMK21 ----- ICSF - Refresh PKA Cryptographic Key Data Set -----
COMMAND ===>
```

Enter the name of the new PKDS below.

```
New PKDS ===> 'FIRST.EMPTY.PKDS'
```

Press ENTER to refresh the PKDS.

Press END to exit to the previous menu

Figure 107. Refresh PKDS

When you press ENTER, the PKDS is refreshed and becomes the in-storage copy.

6. In the New PKDS field, enter the name the initialized PKDS to make it the active PKDS.

Refreshing the CKDS at Any Time

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 10, “Managing Cryptographic Keys by Using the Key Generator Utility Program,” on page 191. For information on using the dynamic CKDS callable services, refer to the *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

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, refer to “Steps for disallowing dynamic CKDS updates during KGUP updates” on page 192.

1. Enter option 2, MASTER KEY, on the ICSF Primary Menu panel to access the Master Key Management Panel.
2. Enter option 1, INIT/REFRESH/UPDATE CKDS to access the Initialize a CKDS panel, which is shown in Figure 108 on page 161.

```

CSFCKD10 ----- ICSF - Initialize a CKDS -----
COMMAND ==>

Enter the number of the desired option.

  1 Initialize an empty CKDS (creates the header and system keys)
    Record authentication required (Y/N)
  2 REFRESH - Activate an updated CKDS

Enter the name of the CKDS below.

CKDS ==> 'PIN1.CKDS'

```

Figure 108. Selecting the Refresh Option on the ICSF Initialize a CKDS Panel

3. In the CKDS field, specify the name of the disk copy of the CKDS that you want ICSF to read into storage.
4. Choose option 2, REFRESH, and press ENTER.
ICSF places the disk copy of the specified CKDS into storage. During a REFRESH, ICSF does not load into storage any partial keys that may exist when you enter keys manually. 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.
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.
5. 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 refresh the CKDS. For information about these other methods, see “Refreshing the In-Storage CKDS” on page 225.

Refreshing the PKDS at any time

When you initialize a PKDS for the first time, you can copy the disk copy of the PKDS 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*. 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.

Note: Prior to refreshing a PKDS, consider temporarily disallowing PKDS write, create and delete services using the ICSF Administrative Control Functions panel.

1. Enter option 2, MASTER KEY MGMT, on the ICSF Primary Menu panel to access the Master Key Management Panel.
2. Enter option 7, REFRESH PKDS to access the Refresh PKA Cryptographic Key Data Set panel, which is shown in Figure 121 on page 173

3. In the New 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 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.
4. Press END to return to the Primary Menu panel.

Reentering master keys when they have been cleared

In these situations, the PCIXCC, CEX2C, or CEX3C clears the master key registers so that the master key values are not disclosed.

- If the PCIXCC, CEX2C, or CEX3C 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 PCIXCC, CEX2C, or CEX3C 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 are still enciphered under the cleared DES or AES master keys. The PKA private keys are also each enciphered under the cleared asymmetric master key. Therefore, to recover the keys in the CKDS, and the PKA private keys, 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: If you are running in PR/SM logical partition (LPAR) mode, there are several situations (listed previously) that can cause loss of master keys and other data. You must then reenter the master keys in each LP. 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 371.

Note: If PPINIT was used initially, you must rerun the utility with the same pass phrase.

When the PCIXCC, CEX2C, or CEX3C clears the master keys, reenter the same master keys by using these steps:

1. Check the status of the PKA callable services. If they are enabled, use the Administrative Control Functions to disable them. See “Steps for enabling and disabling PKA services” on page 168 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 142.

4. When you enter the new DES-MK master key, select option 2, MASTER KEY MGMT, from the primary menu. The Master Key Management panel appears. See Figure 109.

To activate the DES-MK master key you just entered, you need to set it.

5. To set the DES-MK master key, choose option 2 on the panel and press ENTER.

```
CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 2

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                    - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS          - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK                - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS          - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS          - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS             - Activate an updated PKA Cryptographic Key Data Set
```

Figure 109. Selecting the Set Host Master Key Option on the ICSF Master Key Management Panel

When you select option 2, ICSF checks that the states of the registers are correct. ICSF then transfers the DES-MK master key from the new master key register to the master key register. This process sets the DES-MK master key. When ICSF attempts to set the DES-MK master key, it displays a message on the top right of the Master Key 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 DES-MK master key, the DES-MK master key that enciphers the existing CKDS now exists.

6. You can now change the DES-MK master key, if you choose to, for security reasons. Continue with “Steps for changing master keys.”

Steps for changing master keys

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 when you reenter the cleared master keys.

There are three main steps involved in changing the DES-MK master key or AES-MK master key:

1. Enter the DES-MK or AES-MK master key parts.
2. Reencipher the CKDS under the new DES-MK or AES-MK master key.
3. Change the new DES-MK or AES-MK master key and activate the reenciphered CKDS.

Notes:

1. When changing the master key, remember to change the name of the CKDS in the Installation Options Data Set.
2. DES and AES master keys can be changed separately or together.

There are six main steps involved in changing the asymmetric-keys master keys:

1. Disable PKA Services
2. Enter the ASYM-MK.
3. Reencipher the PKDS under the new asymmetric-keys master keys.
4. Refresh the PKDS.
5. Enable PKA Services
6. Enable Dynamic PKDS Access.

Note: When changing the master key, remember to change the name of the PKDS in the Installation Options Data Set.

Symmetric Master Keys and the CKDS

The step-by-step procedure for changing the DES-MK or AES-MK master key, reenciphering the CKDS, and activating the new DES-MK or AES-MK master key are presented in “Steps for changing the DES-MK or AES-MK master key and reenciphering the CKDS” on page 165. This topic provides some background on the contents of the master key registers during the key change process, and some compatibility mode considerations.

A DES-MK or AES-MK master key and a CKDS that contains keys are enciphered under that master key already exist. Prior to replacing this existing DES-MK or AES-MK master key with the new master key, you must reencipher the CKDS under the new DES-MK or AES-MK master key.

Note: Prior to reenciphering a CKDS, consider temporarily disallowing dynamic CKDS update services. For more information, refer to “Steps for disallowing dynamic CKDS updates during KGUP updates” on page 192.

When the new DES-MK or AES-MK master key is placed into the master key register, you must reencipher all disk copies of the CKDS under the new DES-MK or AES-MK master key. Then you are ready to activate the master key. When you change the master key, you have ICSF replace the in-storage copy of the CKDS with the reenciphered disk copy. This also makes the new master key active on the system.

The procedures you use to activate the new master key depend on your system's compatibility mode. 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 PCIXCC, CEX2C, and CEX3C installed, when you start ICSF, you must go to the Master Key Management panel (Figure 109 on page 163) and do a set (option 2). This will change the master keys of all the PCIXCCs, CEX2Cs, and CEX3Cs.

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 changing the DES-MK or AES-MK master key and reenciphering the CKDS

Note: If running in a sysplex, see Chapter 9, “Running in a Sysplex Environment,” on page 181

1. Enter the key parts of the new master key that you want to replace the current master key. For information about how to do this procedure, see “Entering master key parts” on page 135.

The new master key register must be full when you change the master key.

2. Select option 3, REENCIPHER CKDS, on the Master Key Management panel, as shown in Figure 110 on page 166, 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.


```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 3

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                  - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS         - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK               - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS         - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS         - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS           - Activate an updated PKA Cryptographic Key Data Set

```

Figure 110. Selecting the Reencipher CKDS option on the ICSF Master Key Management Panel

3. The Reencipher CKDS panel appears. See Figure 111.

```

CSFCMK10 ----- ICSF - Reencipher CKDS -----
COMMAND ==>

To reencipher all CKDS entries from encryption under the current DES/
Symmetric-keys master key to encryption under the new master key enter
the CKDS names below.

Input CKDS ==> 'CKDS.CURRENT.MASTER'

Output CKDS ==> 'CKDS.NEW.MASTER'

```

Figure 111. Reencipher CKDS

4. 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.

Note: The output data set should already exist although it must be empty. For more information about defining a CKDS, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

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.

6. If you have more than one CKDS on 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 Master Key Management panel.
Changing the master key 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 compatibility or co-existence mode, *do not* select option 4, the Change option. To activate the changed master key when running in compatibility or co-existence mode, you need to re-IPL MVS and start ICSF. When you re-IPL MVS 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 4, CHANGE MK, on the Master Key Management panel.
When you press the ENTER key, the Change Master Key panel appears. See Figure 112.

```

CSFCMK20 ----- ICSF Change Master Key -----
COMMAND ==>>

Enter the name of the new CKDS below:

New CKDS ==>> 'CKDS.NEW.MASTER'

When the master key is changed, the new CKDS will become active.

```

Figure 112. Change Master Key Panel

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 Reencipher CKDS panel, which is shown in Figure 58 on page 122, 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.

You can use a utility program to reencipher the CKDSs and change the master key instead of using the panels. “Reenciphering a disk copy of a CKDS and changing the master key” on page 339 describes how to use the utility program for these procedures.

Asymmetric master keys and the PKDS

The step-by-step procedure for changing the asymmetric master keys is documented in this topic.

When the PKDS is shared by multiple images in a sysplex environment, the asymmetric-key master key must also be changed on all the sharing systems. See Chapter 9, “Running in a Sysplex Environment,” on page 181.

Steps for enabling and disabling PKA services

When you enter or change the ASYM-MK, you must first disable the PKA services. To enable or disable PKA services:

1. Access the administrative control functions by choosing option 4, ADMINCNTL, on the Primary Menu panel, as shown in Figure 113.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 4

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 113. Selecting Administrative Control on the ICSF Primary Menu Panel

The Administrative Control Function panel appears. See Figure 114.

```
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
      -----                               -
. Dynamic CKDS Access                       ENABLED
. PKA Callable Services                     ENABLED
. Dynamic PKDS Access                       ENABLED
```

Figure 114. Enabling and Disabling the PKA Callable Services

2. Enter the appropriate character and press ENTER.
 - To enable the PKA callable services, enter an 'E' before the PKA Callable Services function.

1

- 1

1

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

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

COPROCESSOR  SERIAL NUMBER                               STATUS
-----
_ A06                                     ACTIVE
_ A07                                     ACTIVE
_ E04                                     ACTIVE
_ F03                                     ACTIVE
E X02          42-K0001                               ACTIVE
_ X04          42-K0043                               DEACTIVATED
_ X05          42-K0058                               DISABLED
```

Figure 115. Selecting the coprocessor on the Coprocessor Management Panel

3. Select the coprocessor(s) for asymmetric-key master key entry by entering 'E' before the coprocessor and pressing enter.
4. The Master Key Entry panel appears. See Figure 116 on page 170.

Note: You only RESET if the contents of the registers are not empty.

```

CSFDKE50 ----- ICSF - Master Key Entry -----
COMMAND ==>

                AES new master key register           : EMPTY
                DES new master key register            : EMPTY
                Asymmetric new master key register      : EMPTY

Specify information below
Key Type ==> ____ (AES-MK, ASYM-MK, DES-MK)

Part      ==> FIRST (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 59

Key Value ==> 8F887096A8D4922B
           ==> 75D1189666F4DAA7
           ==> 9B28AEFA8C47760F (AES-MK and ASYM-MK only)
           ==> 0000000000000000 (AES-MK only)

```

Figure 116. The Master Key Entry Panel with First Key Values

5. Fill in the panel

- a. Enter the master key type in the Key Type field.
In this example we are entering ASYM-MK. An asymmetric-key master key requires at least two key parts. You may enter additional key parts if necessary.
- b. Enter FIRST in the Part field.
- c. Enter the two-digit checksum and the three 16-digit key values (if you did not use random number generate).
- d. Make sure you have recorded the three 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 cryptographic coprocessor calculated, the key part is accepted. The message at the top of the panel will now display KEY PART LOADED.
The Asymmetric-keys new master key register status change to PART FULL. The hash pattern that is calculated for the key part appears near the bottom of the panel. Compare it with the pattern generated by the checksum, VP, HP utility or provided by the person who gave you the key part value to enter.
- f. Record the hash pattern.

6. 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.

7. Now enter the FINAL key part.

```
CSFDKE50 ----- ICSF - Master Key Entry -----  
COMMAND ==>  
  
          AES new master key register           : EMPTY  
          DES new master key register           : EMPTY  
          Asymmetric new master key register    : PART FULL  
  
Specify information below  
Key Type ==> ASYM-MK                (AES-MK, ASYM-MK, DES-MK)  
  
Part      ==> FINAL                (RESET, FIRST, MIDDLE, FINAL)  
  
Checksum  ==> 53  
  
Key Value ==> 8FDAD096A8D4922B  
          ==> 75D1189ADAF4DAA7  
          ==> 9B28333A8C47760F (AES-MK and ASYM-MK only)  
          ==> 0000000000000000 (AES-MK only)
```

Figure 117. The Master Key Entry Panel with Final Key Values

8. Fill in the panel

- a. Enter the master key type in the Key Type field.
In this example we are entering ASYM-MK.
 - b. Enter FINAL in the Part field.
 - c. Enter the two-digit checksum and the three 16-digit key values (if you did not use random number generate).
 - d. Make sure you have recorded the three 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 cryptographic coprocessor calculated, the key part is accepted. The message at the top of the panel will now display KEY PART LOADED.
The Asymmetric-keys new master key register status changes to EMPTY.
The hash pattern that is calculated for the key part appears near the bottom of the panel. Compare it with the pattern generated by the checksum, VP, HP utility or provided by the person who gave you the key part value to enter.
 - f. Record the hash pattern.
9. 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.

10. When you have entered the asymmetric-keys master key parts correctly, the asymmetric-keys master key is active when the final key part is loaded. You must then reencipher and refresh the PKDS ("Steps for reenciphering and refreshing the PKDS") and enable PKA callable services "Steps for enabling and disabling PKA services" on page 168. Also enable Dynamic PKDS Access.
11. When changing the master key, remember to change the name of the PKDS in the Installation Options Data Set.

Steps for reenciphering and refreshing the PKDS

When changing the asymmetric-keys master keys, you must reencipher the private keys and trusted block tokens. Reenciphering the PKDS does not refresh the PKDS. The PKDS must be refreshed using the master key management panel or CSFPUTIL. PKA callable services can not be enabled until the reenciphered PKDS is refreshed.

Note: Beginning with HCR7750, LRECL length in the PKDS has increased. You can share the larger PKDS with down-level systems by installing the toleration APAR OA21807. Even with toleration APAR OA21807 installed, however, be aware that reencipherment of a larger PKDS must always be performed on an HCR7750 or later system.

1. To reencipher the PKDS when the ASYM-MK has been changed, go to the Master Key Management panel and select option 6.

Note: Only keys enciphered under the ASYM-MK are reenciphered.

```
CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 6

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                   - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS         - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK               - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS         - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS         - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS           - Activate an updated PKA Cryptographic Key Data Set
```

Figure 118. Selecting the Reencipher PKDS Option on the Master Key Management Panel

The Reencipher PKDS panel appears. In the Input PKDS field, specify the name of the PKDS that you want ICSF to reencipher under the current ASYM-MK.

2. In the Output PKDS field, specify the name of an empty VSAM data set. ICSF places the reenciphered keys in this data set.

```

CSFCMK11 ----- ICSF - Reencipher PKDS -----
COMMAND ==>

To reencipher all PKDS entries from encryption under the old signature/
asymmetric-keys master key to encryption under the current master key, enter
the PKDS names below.

Input PKDS ==> 'PKDS.CURRENT.MASTER'

Output PKDS ==> 'PKDS.NEW.MASTER'

Press ENTER to reencipher the PKDS.
Press END to exit to the previous menu

```

Figure 119. Reencipher PKDS

3. Press enter to reencipher the PKDS. Once successful, you will then want to refresh the PKDS. Return to the Master Key Management panel and select option 7.

```

CSFCMK00 ----- ICSF - Master Key Management -----
OPTION ==> 7

Enter the number of the desired option above.

1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                           - activate an updated Cryptographic Key Data Set
2 SET MK                   - Set a symmetric (DES or AES) master key
3 REENCIPHER CKDS          - Reencipher the CKDS prior to changing a symmetric
                           - master key
4 CHANGE MK                - Change a symmetric master key and
                           - activate the reenciphered CKDS
5 INITIALIZE PKDS          - Initialize or update a PKA Cryptographic
                           - Key Data Set header record
6 REENCIPHER PKDS          - Reencipher the PKA Cryptographic Key Data Set
7 REFRESH PKDS             - Activate an updated PKA Cryptographic Key Data Set

```

Figure 120. Selecting the Refresh PKDS Option on the Master Key Management Panel

4. The Refresh PKDS panel appears. Enter the name of the PKDS that you want ICSF to use. The PKDS must have already been reenciphered under the current asymmetric master key.

```

CSFCMK21 ----- ICSF - Refresh PKA Cryptographic Key Data Set -----
COMMAND ==>

Enter the name of the new PKDS below.

New PKDS ==> 'PKDS.NEW.MASTER'

Press ENTER to refresh the PKDS.
Press END to exit to the previous menu

```

Figure 121. Refresh PKDS

When you press ENTER, the PKDS is refreshed and becomes the in-storage copy.

Steps for 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 PCIX 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 a new DES-MK or AES-MK master key, reencipher a dummy CKDS, and change the master key. You would need to perform this operation twice to ensure that the master key is cleared from the auxiliary (old) master key register. You would also need to reset the asymmetric-keys master keys and process the PCIXCC, CEX2C, and CEX3C master keys.

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.

Steps for adding PCIXCC, CEX2C, or CEX3C coprocessors after initialization

You may need to initialize PCIXCCs, CEX2Cs, and CEX3Cs after system initialization.

Note: Use this procedure if you did not run the Pass Phrase Initialization utility. If you used the utility, see “Steps for adding a PCIXCC, CEX2C, or CEX3C after first time Pass Phrase Initialization” on page 84.

Follow this procedure.

1. Select option 1, COPROCESSOR MGMT, on the Primary Menu panel.
2. The Coprocessor Management panel, as shown in Figure 122, appears.

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

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

COPROCESSOR  SERIAL NUMBER                                STATUS
-----
_ A06
_ A07
_ F03
E X02        42-K0001                                       ONLINE
_ X04        42-K0043                                       DEACTIVATED
_ X05        42-K0058                                       DISABLED
```

Figure 122. Selecting a coprocessor on the Coprocessor Management Panel

3. Select the Coprocessor to be processed by entering 'E' next to the Coprocessor.
4. The Master Key Entry panel appears. See Figure 123 on page 175.


```

CSFDKE50----- ICSF - Master Key Entry -----
COMMAND ==>

          AES new master key register           : EMPTY
          DES new master key register           : EMPTY
          Asymmetric new master key register    : EMPTY

Specify information below
Key Type ==> ____ (AES-MK, ASYM-MK, DES-MK)

Part      ==> _____ (RESET, FIRST, MIDDLE, FINAL)

Checksum ==> 00

Key Value ==> 0000000000000000
          ==> 0000000000000000
          ==> 0000000000000000 (AES-MK and ASYM-MK only)
          ==> 0000000000000000 (AES-MK only)

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 123. The Master Key Entry Panel to Reset Registers

Ensure that the symmetric and asymmetric new master key registers are EMPTY. If they are not, you will need to RESET to clear the contents of the registers to set a new key value.

5. You must now load the DES-MK, AES-MK and ASYM-MK keys into your system.

If you are going to reload your current master keys, you need to know the current master key value and checksum. If you want the PCIXCC, CEX2C, or CEX3C to become ACTIVE after initialization, you MUST enter the same master key values.

Follow the instructions on “Steps for entering the first master key part” on page 142.

6. When all key parts have been loaded, SET the master key. From the Primary Menu panel choose option 2 - Master Key. From the Master Key Management panel, choose option 2 - SET MK.

Chapter 8. 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 on z890, z990, z9, and z10 systems.

Note: z800 and z900 systems require the CCF be available for ICSF and do not support this usage.

A CKDS initialized on a system without coprocessors can not 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 can not be updated to support systems with coprocessors.

Initializing the CKDS at First-Time Startup

The first time you start ICSF, you must:

- Create a cryptographic key data set (CKDS)
- Create a PKA key data set (PKDS)
- The PKDS is required but can not be used for asymmetric key management
- Initialize the CKDS

You only have to initialize a CKDS the first time you start ICSF on a system. 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 “Refreshing the CKDS at Any Time” on page 178.

Steps for initializing a CKDS

1. Select Option 2, MASTER KEY MGMT, on the ICSF Primary Menu panel
2. Select option 1, INIT/REFRESH/UPDATE CKDS and the Initialize a CKDS panel appears.

```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 1

Enter the number of the desired option above.

 1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
 2 SET MK                  - Set a symmetric (DES or AES) master key
 3 REENCIPHER CKDS         - Reencipher the CKDS prior to changing a symmetric
                             master key
 4 CHANGE MK               - Change a symmetric master key and
                             activate the reenciphered CKDS
 5 INITIALIZE PKDS         - Initialize or update a PKA Cryptographic
                             Key Data Set header record
 6 REENCIPHER PKDS         - Reencipher the PKA Cryptographic Key Data Set
 7 REFRESH PKDS           - Activate an updated PKA Cryptographic Key Data Set

```

Figure 124. ICSF Master Key Management Panel

3. 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.

```

CSFCKD10 ----- ICSF - Initialize a CKDS -----
COMMAND ==>

Enter the number of the desired option.

 1 Initialize an empty CKDS (creates the header and system keys)
   Record authentication required (Y/N)
 2 REFRESH - Activate an updated CKDS

Enter the name of the CKDS below.

CKDS ==> 'FIRST.EMPTY.CKDS'

```

Figure 125. ICSF Initialize a CKDS Panel

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*, SA22-7520.

4. Choose option 1, Initialize an empty CKDS, and press ENTER.
To improve performance, answer **N** to Record authentication required.
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.

Refreshing the CKDS at Any Time

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*.

Notes:

1. Prior to refreshing a CKDS, consider temporarily disallowing dynamic CKDS update services.
2. You may refresh any CKDS with the REFRESH CKDS option. This includes CKDS that were initialized on systems with master keys. This is the only way to share a CKDS with a system that has cryptographic coprocessors. If you are sharing a CKDS with encrypted keys, the system with no coprocessors can not manage the encrypted keys.

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.

1. Enter option 2, MASTER KEY, on the ICSF Primary Menu panel to access the Master Key Management Panel.
2. Select option 1, INIT/REFRESH/UPDATE CKDS and the Initialize a CKDS panel appears.

```
CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 1

Enter the number of the desired option above.

  1 INIT/REFRESH/UPDATE CKDS - Initialize a Cryptographic Key Data Set or
                             activate an updated Cryptographic Key Data Set
  2 SET MK                   - Set a symmetric (DES or AES) master key
  3 REENCIPHER CKDS         - Reencipher the CKDS prior to changing a symmetric
                             master key
  4 CHANGE MK               - Change a symmetric master key and
                             activate the reenciphered CKDS
  5 INITIALIZE PKDS         - Initialize or update a PKA Cryptographic
                             Key Data Set header record
  6 REENCIPHER PKDS         - Reencipher the PKA Cryptographic Key Data Set
  7 REFRESH PKDS           - Activate an updated PKA Cryptographic Key Data Set
```

Figure 126. ICSF Master Key Management Panel

3. In the CKDS field, specify the name of the disk copy of the CKDS that you want ICSF to read into storage.

```
CSFCKD10 ----- ICSF - Initialize a CKDS -----
COMMAND ==>

Enter the number of the desired option.

  1 Initialize an empty CKDS (creates the header and system keys)
    Record authentication required (Y/N)
  2 REFRESH - Activate an updated CKDS

Enter the name of the CKDS below.

CKDS ==> 'FIRST.EMPTY.CKDS'
```

Figure 127. ICSF Initialize a CKDS Panel

4. 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.

5. Press **END** to return to the Primary Menu panel.

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 can not be managed in this configuration.

- Key record create (CSNBKRC)
- Key record write (CSNBKRW)
- Key record delete (CSNBKRD)
- Key record read (CSNBKRR)

Key record read will not return a clear key token to the caller unless the caller is in supervisor state or system key.

These services support labels for the key identifier:

- 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)
- One way hash (CSNBOWH)
- MDC generate (CSNBMDG)

Chapter 9. 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.

Attention: If you are running on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC and wish to share your CKDS and PKDS with another system, such as a CCF system on a z900, you should initialize the CKDS and PKDS on the CCF system.

CKDS management in a sysplex

The systems sharing a CKDS may be different LPARs on the same system or different systems across multiple zSeries Processors. One requirement for sharing the CKDS is that the same DES Master Key be installed on all systems sharing that CKDS. It is not required to share the CKDS across a sysplex. Each system may have its own DES Master Key and its own CKDS. A sysplex may have a combination of systems that share a CKDS and individual systems with separate CKDSs.

There is no requirement that the DOMAINS must be the same to share a CKDS.

When sharing the CKDS, a few precautions should be observed:

- Dynamic CKDS services update the DASD copy of the CKDS and the in-storage copy on the system where it is run. The SYSPLEXCKDS option in the ICSF installation options data set provides for sysplex-wide consistent updates of the DASD copy of the CKDS and the in-storage copies of the CKDS on all members of the sysplex sharing the same CKDS. If SYSPLEXCKDS(YES,FAIL(xxx)) is coded in the installation options data set, a sysplex broadcast message will be issued informing sysplex members of the CKDS update and requesting them to update their in-storage CKDS copy. If SYSPLEXCKDS(NO,FAIL(xxx)) is coded in the installation options data set, there is no sysplex broadcast of the update. In order to update the in-storage copy of all images that share the CKDS, you must perform a CKDS REFRESH on each image. This can be done by using either the TSO panels or the CSFEUTIL utility.
- If multiple sysplexes share a CKDS, or if a sysplex and other non-sysplex systems share a CKDS, there is no provision for automatic update of the in-storage copies of the CKDS on the systems which are not in the same sysplex as the system initiating the CKDS update.
- If KGUP is used to update the CKDS, the update is only made to the DASD copy of the CKDS. A manual refresh of the in-storage copy of the CKDS using the ICSF TSO panels or the CSFEUTIL utility is required on all systems sharing the updated CKDS.

Restriction: If you initialized your CKDS on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC, the CKDS cannot be shared with other CCF systems.

Setting DES and AES Master Keys when Sharing a CKDS

Setting master keys for the first time in a sysplex environment is best accomplished by using the Pass Phrase Initialization utility. You need to allocate an empty CKDS and update the options data set on all the LPARs that will be sharing the CKDS. When ICSF is started for the first time, the CKDS needs to be initialized once. All the other LPARs need only to load the same DES master key, and then set the master key.

Note: AES master keys are only supported with:

- FMID HCR7751 running on z9 and z10 servers with a CEX2C and the Nov. 2008 or later licensed internal code (LIC), or
- FMID HCR7770 running on z10 servers.

ICSF releases before HCR7751 do not support secure AES keys and require APAR OA26579 for toleration.

Using Pass Phrase Initialization

Use the Pass Phrase Initialization utility to initialize ICSF in a CKDS shared environment. Start ICSF in the first LPAR. From the first LPAR, follow the instructions in Chapter 5, “Using the Pass Phrase Initialization Utility,” on page 73. Once the LPAR has been successfully initialized, start ICSF in the other LPARS that are sharing the CKDS. Next, from each LPAR that is sharing the same CKDS, go to the Pass Phrase Initialization panel:

- Enter the same exact pass phrase as entered on the first LPAR
- Enter the same exact CKDS name and PKDS name as entered on the first LPAR
- Select 'Reinitialize System' if running on a non-CCF system; if running on a CCF system:
 - Respond **N** to 'Initialize the CKDS and PKDS'
 - Respond to the remaining questions as for the first LPAR

These steps will load and set the same master keys as in the first LPAR and activate the same CKDS.

Using Master Key Entry

You can alternatively use master key entry to set master keys in a sysplex environment. Load your master keys in the first LPAR as described in “Entering master key parts” on page 93 or “Entering master key parts” on page 135. For all subsequent LPARs, enter the master keys as described in “Reentering master keys when they have been cleared” on page 117 or “Reentering master keys when they have been cleared” on page 162.

Changing DES and AES Master Keys when Sharing a CKDS

Refreshing the CKDS when the CKDS is shared

If a CKDS refresh operation is to be performed, dynamic CKDS updates should be disabled on all sysplex systems until the operation has been completed.

Changing master keys should be done with care in a sysplex environment. If a master key change operation is to be performed on a member of the 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. Follow the procedure in “Steps for changing master keys” on page 119 or “Steps for changing master keys” on page 163. Changing the master key and reenciphering the CKDS should be done on an image running the latest level of ICSF. On the other images sharing that CKDS, enter the new master key and then change the master key. Reenciphering the CKDS is not necessary. During the master key change across a sysplex there should not be any applications that pass internal tokens from one image to another.

PKDS management in a sysplex

The systems sharing a PKDS may be different LPARs on the same system or different systems across multiple zSeries and S/390 Processors. The only requirement for sharing the PKDS is that the same PKA 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 PKA Master Keys and its own PKDS. A sysplex may have a combination of systems that share a PKDS and individual systems with separate PKDSs.

When sharing the PKDS, a few precautions should be observed:

- Dynamic PKDS services update the DASD copy of the PKDS and the in-storage copy on the system where it is run. The SYSPLEXPKDS option in the ICSF installation options data set provides for sysplex-wide consistent updates of the DASD copy of the PKDS and the in-storage copies of the PKDS on all members of the sysplex sharing the same PKDS. (Note that all members of the sysplex sharing the PKDS must be running ICSF HCR7751 or higher in order to participate in the sysplex-wide consistency of PKDS data.) If SYSPLEXPKDS(YES,FAIL(xxx)) is coded in the installation options data set, a sysplex broadcast message will be issued informing sysplex members of the PKDS update and requesting them to update their in-storage PKDS copy. If SYSPLEXPKDS(NO,FAIL(xxx)) is coded in the installation options data set, there is no sysplex broadcast of the update. In order to update the in-storage copy of all images that share the PKDS, you must perform a PKDS REFRESH on each image. This can be done by using either the TSO panels or the CSPEUTIL utility.

There is no longer a PKDS cache. ICSF maintains an in-storage copy of the PKDS.

It is highly recommended that the SMK and KMMK be the same on all systems sharing the PKDS in order to reencipher the PKDS when a PKA master key change. PKDS reencipher requires a PCICC, PCIXCC, CEX2C, or CEX3C on your system. PKDS reencipher is not supported on CCF-only systems. For instructions on creating this environment, see “Steps for setting the SMK equal to the KMMK” on page 131.

When sharing the PKDS, this precaution should be observed:

- The PKDS must be initialized for PKA callable services to be enabled. Use the TSO panels to initialize an existing or a new PKDS.

Restriction: The PKDS can be shared between a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system and CCF systems (z900 or z800). However, DSA tokens and RSA tokens encrypted under the KMMK (if KMMK is not equal to the SMK) are not usable on the z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system.

Note: Beginning with HCR7750, LRECL length in the PKDS has increased. You can share the larger PKDS with down-level systems by installing the toleration APAR OA21807. Even with toleration APAR OA21807 installed, however, be aware that reencipherment of a larger PKDS must always be performed on an HCR7750 or later system.

Steps for changing PKA master keys when sharing a PKDS

If you have multiple systems sharing a PKDS and make changes to the PKA master keys, you must reencipher and activate the PKDS. A PCICC, PCIXCC, CEX2C, or CEX3C is required on your system for this process.

Assume you have two systems, A and B sharing a PKDS data set, OLDPKDS. The steps to reencipher and activate are:

1. From SYSTEM A, disable PKA callable services (enter a 'D' prior to the function (see “Steps for enabling and disabling PKA services” on page 124).
2. On SYSTEM B, disable Dynamic PKDS Access (enter a 'D' prior to the function as in Figure 128)

```
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 -----
. Dynamic CKDS Access	ENABLED
. PKA Callable Services	ENABLED
D Dynamic PKDS Access	ENABLED

Figure 128. Administrative Control Functions

3. On system A, change the Master Key (see “PKA master keys and the PKDS” on page 124)
4. On system A, reencipher OLDPKDS, creating NEWPKDS (see “Steps for reenciphering and refreshing the PKDS” on page 129)
5. On system A, refresh NEWPKDS
6. On system A, enable PKA callable services (see “Steps for enabling and disabling PKA services” on page 124).
7. On system A, enable Dynamic PKDS Access (see Figure 128)
8. On system B, disable PKA callable services (see “Steps for enabling and disabling PKA services” on page 124).
9. On system B, change the Master Key (see “PKA master keys and the PKDS” on page 124)
10. On system B, refresh NEWPKDS
11. On system B, enable PKA callable services (see “Steps for enabling and disabling PKA services” on page 124).
12. On system B, enable Dynamic PKDS Access (see Figure 128)

Steps for refreshing 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 automatic update of the in-storage copies of the PKDS on the systems which are not in the same sysplex as the system initiating the PKDS update.

When you are sharing the PKDS in a sysplex, there will be occasions when you change or delete PKDS records, causing changes to the PKDS. In order to reflect the change on other systems in your sysplex, you must refresh the PKDS on each sharing system. From the Master Key Management panel, select option 7 and press enter. There is an additional panel (CSFCMK21) where the PKDS is specified.

```

CSFMKM00 ----- ICSF - Master Key Management -----
OPTION ==> 7

Enter the number of the desired option above.

1 INIT/REFRESH CKDS - Initialize a Cryptographic Key Data Set or activate
                     an updated Cryptographic Key Data Set
2 SET MK             - Set a symmetric (DES or AES) master key
3 REENCIPHER CKDS    - Reencipher the CKDS prior to changing a symmetric
                     master key
4 CHANGE MK          - Change a symmetric master key and
                     activate the reenciphered CKDS
5 INITIALIZE PKDS    - Initialize or update a PKA Cryptographic
                     Key Data Set header record
6 REENCIPHER PKDS    - Reencipher the PKA Cryptographic Key Data Set
7 REFRESH PKDS       - Activate an updated PKA Cryptographic Key Data Set

```

Figure 129. ICSF Master Key Management Panel

Sharing and migrating a CKDS/PKDS between a CCF system and a PCIXCC, CEX2C, or CEX3C system

The z890 and z990 support the PCI X Cryptographic Coprocessor (PCIXCC) and Crypto Express2 Coprocessor (CEX2C).

z9 EC and z9 BC support the Crypto Express2 Coprocessor (CEX2C).

z10 EC and z10 BC support the Crypto Express2 Coprocessor (CEX2C) and the Crypto Express3 Coprocessor (CEX3C).

The z900 and z800 support the Cryptographic Coprocessor Feature (CCF). The PCI Cryptographic Coprocessor (PCICC) is an optional feature.

When sharing a CKDS/PKDS between multiple LPARs, these need to be considered:

1. If mixing z990, z890, z9 EC, z9 BC, z10 EC and z10 BC and legacy systems, the CKDS must have been initialized on the legacy (CCF) system. A CKDS initialized on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system cannot be shared with a CCF system; ICSF will not start.
2. The DES-MK on your PCIXCC, CEX2C, or CEX3C must match the DES master key on the CCF.
3. The ASYM-MK on your PCIXCC, CEX2C, or CEX3C system must match the SMK master key on the CCF system. If mixing different releases of ICSF, make sure service is up to date with regard to CKDS/PKDS toleration.
If sharing a PKDS between z990, z890, z9 EC, z9 BC, z10 EC and z10 BC and a legacy system, and the legacy system does NOT have the SMK=KMMK, then the PKDS needs to be initialized on the legacy system. If not, the KMMK hash will not be in the PKDS header and PKA Callable Services cannot be enabled.
4. Retained keys on the PCICC, PCIXCC, CEX2C, or CEX3C cannot be shared across LPARs. Retained keys are domain specific; they can only be used on the domain where they were generated.

Note: ICSF needs to be started to perform the PKDS Initialization.

CCF only system

SMK equal to KMMK

- Using Master Key Entry
 1. Start ICSF on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system, pointing to the initialized CKDS/PKDS. You will see the message: CSFM419E INCORRECT MASTER KEY (BOTH) ON PCI X CRYPTOGRAPHIC COPROCESSOR Xnn, SERIAL NUMBER nnnnnnnn.
 2. Using Master Key Entry, load the value of the CCF DES master key into the new DES-MK register. Load the value of the CCF SMK/KMMK master key into the new ASYM-MK register. You will need the checksums for each of these values.
 3. Set the DES master key.
 4. Enable PKA Callable Services/Dynamic PKDS Access.
- Using Pass Phrase Initialization
 1. Start ICSF on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system, pointing to the initialized CKDS/PKDS.
 2. Using PPINIT, type in the same pass phrase used to initialize CCF system. Respond N to Initialize the CKDS/PKDS? (Y/N) question.

SMK not equal to KMMK

Without a PCICC, the PKDS reencipher must run on the PCIXCC, CEX2C, or CEX3C. If it is not, the z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system will not be able to use the tokens encrypted under the KMMK. This procedure requires that you switch between your legacy and z990/z890 TSO sessions.

- Using Master Key Entry

It does not matter whether you reencipher to the KMMK or the SMK. This checklist reenciphers to the SMK.

 1. Start ICSF on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system, pointing to the initialized CKDS/PKDS.
 2. Define an empty PKDS.
 3. Load the value of the CCF DES master key into the new DES-MK register. You will need the checksum.
 4. Load the value of the CCF KMMK master key into the new ASYM-MK register. You will need the checksum. The ASYM-MK is automatically set when the final key part is loaded.
 5. Load the value of the CCF SMK master key into the new ASYM-MK register. You will need the checksum. The ASYM-MK is automatically set when the final key part is loaded. The old ASYM-MK register now contains the KMMK value and the current ASYM-MK register contains the SMK value.
 6. Set the DES-MK.
 7. Reencipher the active PKDS to the empty PKDS.
 8. Refresh the new PKDS. Enable PKA Callable Services/Dynamic PKDS Access.
 9. Update options dataset to point to the new PKDS.
 10. On CCF system, disable PKA Callable Services.
 11. Reset the KMMK register.
 12. Load the value of the CCF SMK master key into the KMMK register.
 13. Activate the new PKDS.
 14. Enable PKA Callable Services/Dynamic PKDS Access.

15. Update options dataset to point to the new PKDS.
- Using Pass Phrase Initialization
 1. On a CCF system, use PPKEYS to get the clear key values of the SMK and KMMK from a pass phrase. You will need the checksum for each of these values.
 2. On z990/z890, z9 EC, z9 BC, z10 EC and z10 BC system, start ICSF pointing to initialized CKDS/PKDS.
 3. Define an empty PKDS.
 4. Using Master Key Entry, load the value of the CCF KMMK master key into the new ASYM-MK register. You will need the checksum. Load a final key part of zeroes. The ASYM-MK is automatically set when the final key part is loaded.
 5. Using PPINIT, type in the pass phrase used to initialize the CCF system, enter the names of the initialized CKDS/PKDS, respond N to Initialize the CKDS/PKDS? (Y/N).
 6. Reencipher the PKDS to the empty PKDS.
 7. Refresh the new PKDS.
 8. Update options dataset to point to new PKDS.
 9. On a CCF system, disable PKA Callable Services.
 10. Using Master Key Entry, reset the KMMK register.
 11. Load the value of the SMK into the KMMK register. You can get the clear key value of the SMK using the PPKEYS utility. You will need the SMK checksum.
 12. Activate the new PKDS.
 13. Enable PKA Callable Services/Dynamic PKDS Access.
 14. Update options dataset to point to new PKDS.

CCF with PCICCs

SMK equal to KMMK

- Using Master Key Entry
 1. Start ICSF on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system, pointing to the initialized CKDS/PKDS. You will see message: CSFM419E INCORRECT MASTER KEY (BOTH) ON PCI X CRYPTOGRAPHIC COPROCESSOR Xnn, SERIAL NUMBER nnnnnnnn.
 2. Using Master Key Entry, load the value of the CCF DES master key into the new DES-MK register. Load the value of the CCF SMK/KMMK master key into the new ASYM-MK register. You will need the checksums for each of the master key values.
 3. Set the DES master key.
 4. Enable PKA Callable Services/Dynamic PKDS Access.
- Using Pass Phrase Initialization
 1. Start ICSF on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system, pointing to the initialized CKDS/PKDS.
 2. Using PPINIT, type in the same pass phrase used to initialize CCF system. Respond N to Initialize the CKDS/PKDS? (Y/N).

SMK not equal to KMMK

Make the SMK=KMMK prior to sharing the CKDS/PKDS with the z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system.

- Using Master Key Entry
 1. Define an empty PKDS.
 2. On the CCF system, disable PKA Callable Services.
 3. Using Master Key Entry, reset ALL-PKA registers. Load the value of the CCF KMMK master key into the SMK/KMMK/ASYM-MK registers on all CCF/PCICC coprocessors. You will need the checksum. The ASYM-MK is automatically set when the final key part is loaded.
 4. Reencipher the PKDS to the empty PKDS.
 5. Activate the new PKDS.
 6. Enable PKA Callable Services/Dynamic PKDS Access.
 7. Update options dataset to point to new PKDS.
 8. Start ICSF on the z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system, pointing to initialized CKDS/PKDS.
 9. Load the value of the CCF DES master key into the new DES-MK register.
 10. Load the value of the CCF KMMK master key into the new ASYM-MK register. You will need the checksum. The ASYM-MK is automatically set when the final key part is loaded. The current ASYM-MK now has the same value as the SMK/KMMK/ASYM-MK on the CCF/PCICC(s).
 11. Set the DES-MK.
 12. Enable PKA Callable Services/Dynamic PKDS Access.
- Using Pass Phrase Initialization
 1. On the CCF system, use PPKEYS to get the clear key values of the SMK and KMMK from a pass phrase. You will also need the checksum for each of these values.
 2. Define an empty PKDS. Disable PKA Callable Services.
 3. Using Master Key Entry, load the value of the CCF KMMK master key into the new ASYM-MK register on the PCICC(s). You will need the checksum. Load a final key part of zeroes. The ASYM-MK is automatically set when the final key part is loaded. The current ASYM-MK is now the same as the KMMK value.
 4. Load the value of the CCF SMK into the new ASYM-MK register on the PCICC(s). You will need the checksum. Load a final key part of zeroes. The ASYM-MK is automatically set when the final key part is loaded. The current ASYM-MK is now the same as the SMK value. The KMMK value is now in the old ASYM-MK register.
 5. Reset the KMMK register on the CCFs. Load the SMK value into the KMMK register. Now the KMMK = SMK.
 6. Reencipher the PKDS to the empty PKDS.
 7. Activate the new PKDS.
 8. Enable PKA Callable Services/Dynamic PKDS Access.
 9. Update options dataset to point to the new PKDS.
 10. Start ICSF on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC system, pointing to the initialized CKDS/PKDS (the one just reenciphered previously).
 11. Using PPINIT, type in the same pass phrase used to initialize CCF system. Respond N to Initialize the CKDS/PKDS? (Y/N).

TKDS management in a sysplex

The systems sharing a TKDS may be different LPARs on the same system or different systems across multiple zSeries processors. 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. When sharing the TKDS, a few precautions should be observed:

- Dynamic TKDS services update the DASD copy of the TKDS and the in-storage copy on the system where it runs. The SYSPLEXTKDS option in the ICSF installation options data set provides for sysplex-wide consistent updates of the DASD copy of the TKDS and the in-storage copies of the TKDS on all members of the sysplex sharing the same TKDS.

If SYSPLEXTKDS(YES,FAIL(xxx)) is coded in the installation options data set, a sysplex broadcast message will be issued informing sysplex members of the TKDS update and requesting them to update their in-storage TKDS copy. If SYSPLEXTKDS(NO,FAIL(xxx)) is coded in the installation options data set, there is no sysplex broadcast of the update.

- If multiple sysplexes share a TKDS, or if a sysplex and other non-sysplex systems share a TKDS, there is no provision for 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.

Chapter 10. Managing Cryptographic Keys by 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, and transport keys. If you are running a z890, z990, z9 EC, z9 BC, z10 EC and z10 BC KGUP supports double length MAC and MACVER keys. Although ANSI transport keys are stored in the CKDS, KGUP does not support the generation or import of ANSI transport keys. KGUP does not support non-standard CV keys.

Restriction: KGUP does not support DATA LAT keys on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC.

To run KGUP, ICSF must be active, master keys must be loaded on the cryptographic coprocessors, the user must have access, and the CKDS must be initialized.

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 194.

Each key that KGUP generates (except clear key value data-encrypting keys and clear AES keys) exists in the CKDS enciphered under your system’s master key. When the master key enciphers a key, the cryptographic facility exclusive ORs the master key with a pattern of characters called a control vector. A master key exclusive ORed with a control vector is called a master key variant.

A unique control vector exists for each type of key the master key enciphers. The cryptographic facility exclusive ORs the master key with the control vector associated with the type of key the master key will encipher. The control vector ensures that a key is only used in the cryptographic functions for which the key is intended. For example, the control vector for an input PIN encryption key ensures that such a key can be used only in PIN translate and PIN verification functions.

When you specify to KGUP to generate an input PIN-encrypting key, the cryptographic facility creates a master key variant for the key. The master key variant is a product of exclusive ORing the master key with the control vector associated with an input PIN-encrypting key. This master key variant enciphers the input PIN-encrypting key so the input PIN-encrypting key is in operational form. KGUP places the input PIN-encrypting key in a CKDS entry.

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 218.

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 KGUP 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 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. Choose option 4, Administrative Control Functions, on the Primary Menu Panel, as shown in Figure 130 on page 193.

```

CSF@PRIM ---- Integrated Cryptographic Service Facility -----
OPTION ==> 4

Enter the number of the desired option.

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

        Licensed Materials - Property of IBM

        5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
        US Government Users Restricted Rights - Use, duplication or
        disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 130. Selecting the Administrative Control Option on the Primary Menu Panel

The Administrative Control Functions panel appears. See Figure 131.

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 131. 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:

- Data-encrypting (DATA) and data-translation (DATAXLAT) keys
- Importer key-encrypting key and exporter key-encrypting key
- Input PIN-encrypting key and output PIN-encrypting key
- MAC generation key and MAC verification key
- PIN generation key and PIN verification key

Restriction: DATAXLAT keys are only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900.

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 DATA key for inclusion into the CKDS and export a copy of the key as either a DATA key or a DATAXLAT key. If you export the copy of the DATA key as a DATA key, the receiver of the key can use it to decipher data. If you export the copy of the DATA key as a DATAXLAT key, the receiver can use the key only to translate cipher text from one DATAXLAT key to another. The receiver of the DATAXLAT 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.

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 251.

The cryptographic facility exclusive ORs a transport key with a control vector when using the transport key to encipher a key. A transport key exclusive ORed with a control vector is called a transport key variant. ICSF uses the control vector associated with the key type that the transport key will encipher. The control vector ensures that when another site imports the key, the resulting operational key can only be the type that the control vector indicates. For example, the control vector for a PIN verification key ensures that the system that receives the key can import the key only as a PIN verification key.

When KGUP generates a PIN generation key, the program generates a key value to create a PIN verification key. You can specify that the key value be an encrypted key value. When you do this, ICSF exclusive ORs the transport key with the control vector for a PIN verification key to create the transport key variant. Then the cryptographic facility enciphers the PIN verification key under the transport key variant.

To view the specific control vector value that is associated with each type of key to create master key variants and transport key variants, see Appendix B, “Control Vector Table.”

Transport key variants ensure that the receiving system uses the key as the type of key that the sending system intended. However transport key variants can only be used if both systems recognize transport key variants. You should use transport key variants when exchanging keys with the 4758 PCI Cryptographic Coprocessor. However, systems with some cryptographic products, such as PCF, do not recognize control vectors. When you exchange keys with such a system, a key that you send or receive is enciphered under a transport key rather than a transport key variant. You just specify to KGUP that the transport key should not be exclusive ORed with a control vector.

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 253 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 can not 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.

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 DATA, DATAxLAT, MAC, MACVER, DATAM, DATAMV, and NULL keys.
- For compatibility with Version 1 Release 1 function, transport and PIN keys can have duplicate labels for different key types. Keys that use the dynamic CKDS update services to create or update, however, must have unique key labels.
- 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 can not 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.

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 132 on page 198 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]

[KEY(key-value[,ikey-value])]

```

Figure 132. 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 196.

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 196.
- 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 CLRAES, CLRDES,

DATA, DATAXLAT, MAC, MACVER, DATAM, DATAMV, and NULL key types, KGUP allows only one key per label. For all other key types, you can have keys with the same labels but different key types.

You can specify any of these key types:

CLRAES

Clear Encryption/decryption key for AES

CLRDES

Clear Encryption/decryption key

DATA Encryption/decryption key

DATAXLAT

Cipher text translate key – DATAXLAT is only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900.

DATAM

Double-length MAC generation key

DATAMV

Double-length MAC verification key

EXPORTER

Exporter key-encrypting key

IMPORTER

Importer key-encrypting key

IPINENC

Input PIN encryption key

MAC Single-length MAC generation key

Note: On a z990, z890, z9 EC, z9 BC and IBM System z10 Enterprise Class, MAC is a single or double-length key.

MACVER

Single-length MAC verification key

Note: On a z990, z890, z9 EC, z9 BC and IBM System z10 Enterprise Class, MACVER is a single or double-length key.

NULL Used to create a null CKDS entry

OPINENC

Output PIN encryption key

PINGEN

PIN generation key

PINVER

PIN verification key

All these types of keys are stored in the CKDS.

Note: For compatibility with previous releases of OS/390 ICSF, KGUP stores internal versions of DATAM and DATAMV keys in the CKDS under the key types of MACD and MACVER, respectively.

ALGORITHM (DESIAES)

This keyword defines the algorithm of the key you are generating. DES is the default value. All key types except CLRAES and CLRDES are valid with the DES value. Only the DATA and NULL key types are valid with AES. Generated operational keys will be encrypted under the respective master key.

Note: To use AES, you need to have an AES-MK

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. You cannot specify an OUTTYPE when you have chosen either CLRAES, CLRDES, DATAMV, PINVER, MACVER, or NULL for the key TYPE.

Refer to Table 12 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 12. Default and Optional OUTTYPES Allowed for Each Key TYPE

TYPE	OUTTYPE (Default)	OUTTYPE (Allowed)
CLRAES	Not Allowed	Not Allowed
CLRDES	Not Allowed	Not Allowed
DATA	DATA	DATA, DATAXLAT*
DATAXLAT	DATAXLAT	DATAXLAT*
DATAM	DATAMV	DATAM, DATAMV
DATAMV	Not Allowed	Not Allowed
EXPORTER	IMPORTER	IMPORTER
IMPORTER	EXPORTER	EXPORTER
IPINENC	OPINENC	OPINENC
MAC	MACVER	MAC, MACVER
MACVER	Not Allowed	Not Allowed
NULL	Not Allowed	Not Allowed
OPINENC	IPINENC	IPINENC
PINGEN	PINVER	PINVER
PINVER	Not Allowed	Not Allowed

Notes:

1. * DATAXLAT is only supported on the IBM @server zSeries 800 and the IBM @server zSeries 900
2. There is no defined OUTTYPE for the AES algorithm and the keyword may not be used with ALGORITHM(AES).

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.

When KGUP generates a key, the program enciphers the key under a master key variant. 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 under a transport key or transport key variant. On the control statement, use the TRANSKEY keyword to specify the transport 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 DATA and MAC keys.

When you input a key value that is in importable form, the key that is specified by the KEY keyword is enciphered under a transport 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 a pair of control statements to create the complementary pair of keys that are needed at the two sites.

Note: You cannot specify two transport keys that were installed without control vectors. 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.

Note: TRANSKEY is not valid with ALGORITHM(AES).

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 transport key variants, 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 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.

Note: To create keys for NOCV processing, NOCV-Enablement keys must exist. For a description of how to create NOCV-Enablement keys, see “Initializing the CKDS and PKDS at First-Time Startup” on page 110.

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 or SINGLE

LENGTH indicates the length of the key to generate. LENGTH(8) generates a single-length key. LENGTH(16) generates a double-length DES key or 128-bit AES key, LENGTH(24) generates a triple-length DES key (DATA only) or a 192-bit AES key. LENGTH(32) generates a 256-bit AES key. If a LENGTH is specified when generating DATAM or DATAMV keys, it must be LENGTH(16). For CLRDES, valid values for the LENGTH keyword are 8, 16, and 24. For CLRAES, valid values for the LENGTH keyword are 16, 24 and 32.

For double-length key types, LENGTH(8) or SINGLE in an ADD or UPDATE statement causes KGUP to generate a double-length key with both halves the same. On the KGUP panel, you can achieve this by specifying 8 in the LENGTH field for a double-length key type.

In either 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. Although the LENGTH keyword is valid when you create control statements to generate DATA keys, KGUP ignores it for DATAXLAT keys. KGUP automatically generates them as single-length keys.

DES

This keyword is no longer supported but is tolerated.

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.

This keyword is required when you specify either DATAMV, MACVER, or PINVER for the TYPE keyword. Because KGUP cannot generate PIN verification or MAC verification keys in operational form, you must always supply values for these types of keys.

For DATAXLAT, supply one key value. DATAXLAT is a single-length key, if you supply a second key value, KGUP discontinues processing the control statement and issues an error message.

For a double-length key (EXPORTER, IMPORTER, IPINENC, OPINEC, PINGEN, PINVER), supply two key values. If you supply only one key value, KGUP will duplicate the key value as the secode key value. KGUP concatenates these two identical values, and then stores and uses the key as if the key was double-length.

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.

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 13 shows the keyword combinations permitted on ADD and UPDATE control statements.

Table 13. Keyword Combinations Permitted in ADD and UPDATE Control Statements

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 ⁴	Yes ¹
UPDATE	Yes	Yes	Yes ¹	Yes ²	Yes ³	Yes ⁴	Yes ¹

Notes:

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.
4. OUTTYPE, TRANSKEY and NOCV are not valid with ALGORITHM(AES). There are no restrictions with ALGORITHM(DES).

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,ikey-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
```

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)
```

For the CLRDES and CLRAES key types, the CLEAR keyword is not allowed. The value in the KEY keyword is the clear key value that will be inserted into the token. Special secure mode is not required for these key types:

```
ADD or UPDATE LABEL(label) TYPE(clrdes)  
CLEAR KEY(key-value)
```

Import an Encrypted Key Value: 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 preferable 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 133. 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 196.

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 134. 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 196.

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 196.
- 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,dataxlat,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 135. 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 PCIXCC, CEX2C, or CEX3C 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. This keyword is only supported on a z990, z890, z9 EC, z9 BC and IBM System z10 Enterprise Class.

The OPKYLOAD control statement has this syntax:

OPKYLOAD

```
LABEL (key-label)
SERNBR (coprocessor-serial-number)
[NOCV]
```

Figure 136. OPKYLOAD Control Statement Syntax

LABEL (key-label)

This label must match the label used to create the key by the TKE workstation on the PCIXCC.

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 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. It also checks that there are no DATA, DATAXLAT, DATAM, DATAMV, MAC, MACVER, or NULL key entries with that label. Each of these keys requires 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 variant for an importer 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. IMPORTER, DATA, DATAM, 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(ATMBRANCH5M0001) CLEAR
```

KGUP checks that an entry with the label ATMBRANCH5M0001 with the type EXPORTER does not already exist in the CKDS. It also checks that there are no DATA, DATAXLAT, DATAM, DATAMV, MAC, MACVER, or NULL key entries with that label. Each of these keys requires 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 variant for an exporter 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(ATMBRANCH5M0001) 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(LOCT0JWL.JULY03) TRANSKEY(SENDJWL.JULY03)
```

KGUP checks that an entry with the label LOCT0JWL.JULY03 for an input PIN-encrypting key does not already exist in the CKDS. It also checks that there are no DATA, DATAM, DATAMV, MAC, MACVER, or NULL key entries with that label. Each of these keys requires 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 variant for an input PIN-encrypting 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 an 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(6709E5593933DA00,1449A3D9ED0A1586)
```

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 BRANCH5M0001 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)
```

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 a PCIXCC, CEX2C, or CEX3C. 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 PCIXCC, CEX2C or CEX3C 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 PCIXCC, 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.

```
OPKYL0AD 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 PCIXCC, CEX2C, or CEX3C 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 control statement with CLRDES keyword

This example shows a control statement that adds a CLRDES key to the CKDS with a random 8 byte key.

```
ADD TYPE(CLRDES) LENGTH(8) LAB(CLRDES.KEYLN8)
```

Example 14 – ADD control statement to add a group of CLRDES keys

This example shows a control statement that adds a group of CLRDES keys to the CKDS. Key value is generated.

```
ADD TYPE(CLRDES) LENGTH(8) LAB(A.CLRDES.KEYLN8,B.CLRDES.KEYLN8,C.CLRDES.KEYLN8)
```

Example 15 – ADD control statement to add a group of CLRDES keys

This example shows a control statement that adds a group of CLRDES keys. The clear key value is specified.

```
ADD TYPE(CLRDES) KEY(2C2C2C2C2C2C2C2C,1616161616161616),  
LAB(X.CLRDES.KEYLN16,Y.CLRDES.KEYLN16,Z.CLRDES.KEYLN16)
```

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.

```
473 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) 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)
```

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) 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. It has a fixed logical record length (LRECL) of 252 bytes.

Programming Interface information

The records in the 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 HHMMSSSTH.

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 HHMMSSSTH.

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 first record in the CKDS is a header record. 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 HHMMSSSTH.

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 HHMMSSSTH.

Sequence number

(Character length 2 bytes) Initially binary zero, incremented 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. 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 “Algorithm for calculating a verification pattern” on page 367.

DES master key authentication pattern

(Character length 8 bytes) The DES master key authentication pattern.

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 that is described in “Algorithm for calculating an authentication pattern” on page 367.

Whenever you start ICSF, ICSF uses the authentication pattern to verify that the current master key is the master key that enciphers the current CKDS. ICSF fails if the authentication pattern that is stored in the CKDS and the authentication pattern that ICSF calculates at startup do not match.

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 that is described in “Algorithm for calculating an authentication pattern” on page 367.

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.

End of Programming Interface information

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 196.

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 137 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 137. 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) of 208 bytes.

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.

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 64 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 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 138. 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 218 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. You must specify SSM(YES) in the installation options data set.

For CCF Systems, if you use logical partition (LPAR) mode, you also need to enable special secure mode on the Change LPAR Crypto panel from the Hardware Master Console of the server support element. If you have the optional TKE workstation, you can use it to enable and disable special secure mode.

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 *MVS Batch Local Shared Resources*.

VSAM provides a deferred write option on VSAM ACB processing when a program uses shared resources. For more information about VSAM processing, see *MVS/DFP Managing VSAM Data Sets* and the *MVS/ESA Data Administration: Macro Instruction Reference*.

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 138 on page 223.

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 the 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 when 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 for Catalogs*, SC26-7394.

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 current CKDS using the ICSF panels” on page 250. 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 341 for details.

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 Primary Menu panel as shown in Figure 139.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 8

Enter the number of the desired option.

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

Figure 139. Selecting the KGUP Option on the Primary Menu Panel

The Key Administration panel appears. See Figure 140 on page 227.

```

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 140. 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 141, 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 141. Selecting the Create Option on the Key Administration Panel

The KGUP Control Statement Data Set Specification panel appears. See Figure 142 on page 228.

```
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 142. 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 14 for a list of these options and the steps to follow for each.

Table 14. 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 146 on page 231. The new control statements will be appended when any existing control statements in the data set.

Table 14. 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 144 on page 230. 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 144 on page 230 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 146 on page 231. 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 145 on page 230. 2. Enter the necessary information to allocate a new data set and press ENTER. The KGUP Control Statement Menu appears. See Figure 146 on page 231. The new control statements will be stored in the new data set.

Figure 143 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 143. 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 144.

```

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 144. Member Selection List Panel

If you specify a new data set name, the Allocation panel appears. See Figure 145.

```

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 145. 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 146 on page 231. 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.


```

CSFCM00 ----- 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 146. 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 238.
 - To create a SET control statement, select option 3. For information, see “Steps for creating a SET control statement” on page 240.
 - To edit the input control statement data set, select option 4. For information, see “Steps for editing control statements” on page 242.

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 147 on page 232.

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 147. Create ADD, UPDATE, or DELETE Key Statement Panel

- 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 197. See that topic for more information about the control statement keywords.
- 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 148 on page 233, 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 203.

```

----- 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 148. 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 149 on page 234.

```

CSFCSE12----- ICSF - Key Type Selection Panel ---- ROW 1 TO 13 OF 11
COMMAND ===>                                     SCROLL ===> PAGE

Select one key type only
KEY TYPE      DESCRIPTION

CLRAES        Clear AES Encryption/decryption key
CLRDES        Clear Encryption/decryption key
DATA          Encryption/decryption key
DATAM         Double-length MAC generation key
DATAMV        Double-length MAC verification key
DATAXLAT      Data-translation key
s EXPORTER    Export key-encrypting key
IMPORTER      Import key-encrypting key
IPINENC       Input PIN-encrypting key
MAC           Message authentication key
MACVER        Message verification key
NULL          Dummy CKDS records
OPINENC       Output PIN-encrypting key
PINGEN        PIN generation key
PINVER        PIN verification key
*****BOTTOM OF DATA*****

```

Figure 149. 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 149, 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 150 on page 235.

```

----- 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 150. 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.

- 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 15. Selecting Range and Label Options

Option	Steps
To have KGUP process only one key label	- Specify the key label in the Label field. - Type NO in the Group Labels field.
To have KGUP process more than one key label	- Specify the first label in the Label field. - Type YES in the Group Labels field.

- 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 16. 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 N0 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 decrypt the key in the Transport Key Label field. 2. Type N0 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 N0 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 N0. If you type N0 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 N0. 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 151 on page 237.

```

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 151. 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 N0 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 152 on page 238.

```

----- 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 152. 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 153.

```

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 153. Selecting the Rename Option on the KGUP Control Statement Menu Panel

2. See Figure 154. 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 209. 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 154. 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 155.

```
CSFCSE12----- ICSF - Key Type Selection Panel ----- ROW 1 To 13 OF 11
COMMAND ===>                                     SCROLL ===> PAGE

Select one key type only
KEY TYPE      DESCRIPTION

CLRAES        Clear AES Encryption/decryption key
CLRDES        Clear Encryption/decryption key
DATA          Encryption/decryption key
DATAM         Double-length MAC generation key
DATAMV        Double-length MAC verification key
DATAXLAT      Data-translation key
s EXPORTER    Export key-encrypting key
IMPORTER      Import key-encrypting key
IPINENC       Input PIN-encrypting key
MAC           Message authentication key
MACVER        Message verification key
NULL          Dummy CKDS records
OPINENC       Output PIN-encrypting key
PINGEN        PIN generation key
PINVER        PIN verification key
*****BOTTOM OF DATA*****
```

Figure 155. 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 155, 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 156.

```
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 156. 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 157.

```
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 157. Selecting the Set Option on the KGUP Control Statement Menu Panel

2. See Figure 158. From this panel you can create a SET control statement. For information about the SET control statement keywords, refer to “Syntax of the SET Control Statement” on page 210.

```
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 158. 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 159 on page 242.

```

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 159. 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 160.

```

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 160. 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 161 on page 243.

```

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(SENDBRANCH5JUNE99)
000006     LABEL(ATMBRANCH5M0001)
000007 /* EXPORT TEST KEY RENAMED */
000008 RENAME LABEL(JWL@SSIDEC97,JWL@SSIJUNE99) TYPE(EXPORTER)
***** ***** BOTTOM OF DATA *****

```

Figure 161. 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 162 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(SENDBRANCH5JUNE99)
000006     LABEL(ATMBRANCH5M0001)
000007 /* EXPORT TEST KEY RENAMED */
000008 RENAME LABEL(JWL@SSIDEC97,JWL@SSIJUNE99) TYPE(EXPORTER)
***** ***** BOTTOM OF DATA *****

```

Figure 162. 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 163 on page 244, 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 163. Selecting the Specify Data Set Option on the Key Administration Panel

The Specify KGUP Data Sets panel appears. See Figure 164.

```

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 = CSFSTMT)
Data Set Name ==> _____
Volume Serial ==> _____ (if uncataloged)

Press ENTER to set the data set names. Press END to exit to the previous panel.

```

Figure 164. 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 CKDS and PKDS at First-Time Startup” on page 110. 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 142 on page 228 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 218.

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 165 on page 246.

```

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 165. 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 166, 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 166. Invoking KGUP by Selecting the Submit Option on the Key Administration Panel

See Figure 167. 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 167. 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 168 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 165 on page 246.
 - 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 168. 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 168 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 168 on page 248, 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 169 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=('NOSM')
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=TS0001,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 169. 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 current 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 243.

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.

1. To access the panels to refresh the current CKDS, choose option 4 on the Key Administration panel, as shown in Figure 170.

```
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 170. Selecting the Refresh Option on the Key Administration Panel

The Refresh in-storage CKDS panel appears. See Figure 171 on page 251.

```

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 171. 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 165 on page 246 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.
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 172.

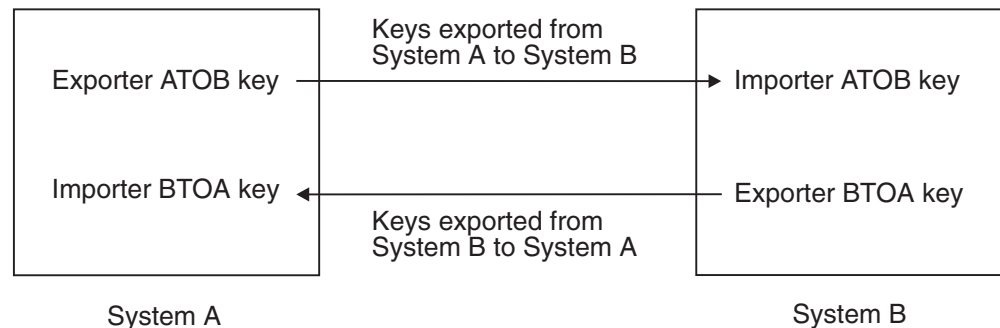


Figure 172. Key Exchange Establishment between Two ICSF Systems

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 decipheres 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.

Notes:

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 173.

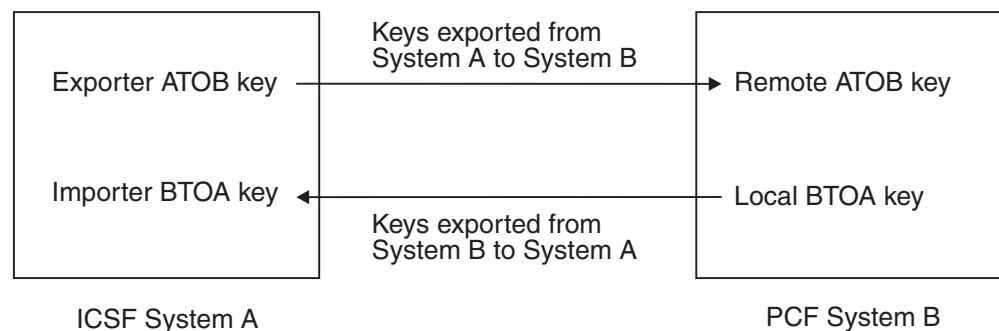


Figure 173. Key Exchange Establishment between an ICSF System and a PCF System

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 4758 PCI Cryptographic Coprocessor Establishing Initial Transport Keys

This scenario describes how an ICSF system and a 4758 PCI Cryptographic Coprocessor establish initial transport keys between themselves. They establish two pairs of complementary importer and exporter keys at each location, as shown in Figure 174.

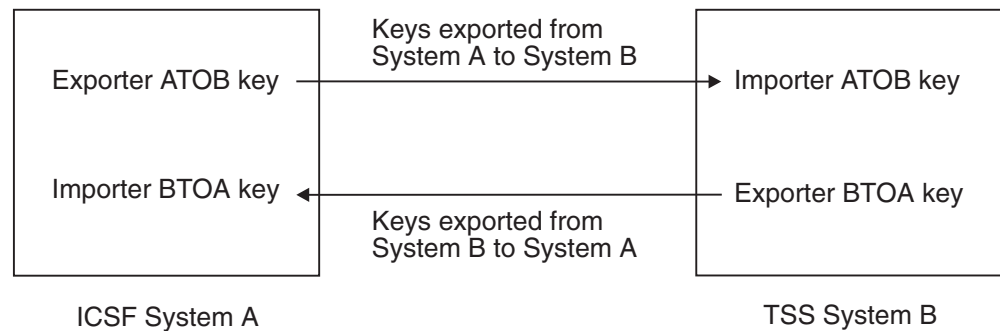


Figure 174. Key Exchange Establishment between a 4758 PCI Cryptographic Coprocessor System and an ICSF System

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.

Notes:

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 11. Viewing and Changing System Status

This topic describes:

- “Displaying administrative control functions”
- “Displaying coprocessor or accelerator status - CCF, PCICC, PCICA” on page 259
- “Displaying coprocessor or accelerator status - PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A” on page 261
- “Changing coprocessor or accelerator status - CCF, PCICC, and PCICA” on page 264
- “Changing coprocessor or accelerator status - PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A” on page 264
- “Displaying coprocessor hardware status - CCF and PCICC” on page 266
- “Displaying coprocessor hardware status - PCIXCC, CEX2C, and CEX3C” on page 273
- “Displaying installation options” on page 280
- “Displaying PCICC default roles” on page 287
- “Displaying PCIXCC, CEX2C, and CEX3C default roles” on page 289
- “Displaying installation exits” on page 293
- “Displaying installation-defined callable services” on page 301

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 PCICC, PCIXCC, CEX2C, and CEX3C coprocessors and PCICA, CEX2A, and CEX3A 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.

Displaying administrative control functions

To display administrative control functions:

1. Select option 4, ADMINCNTL, on the primary menu panel.

```

CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 4

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 175. Primary Panel

The Administrative Control panel appears, which is shown in Figure 176.

```

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
      -----                               -
.  Dynamic CKDS Access                      ENABLED
.  PKA Callable Services                   ENABLED
.  Dynamic PKDS Access                      DISABLED

```

Figure 176. Administrative Control Functions Panel

On this panel, 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.

Note: Disabling PKA callable services also disables PKDS Read and Write access.

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.

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 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 disabled.

Note: Access to the functions performed using this panel can be controlled by setting up profiles in the CSFSERV class for both CSFRSWS and CSFSSWS.

Displaying coprocessor or accelerator status - CCF, PCICC, PCICA

Use the ICSF panels to view the status of the coprocessors or accelerators. To display the status:

1. Select option 1, COPROCESSOR MGMT, on the Primary Option panel, as shown in Figure 177.

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

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 177. Selecting Coprocessor Status on the Primary Menu Panel

2. The Coprocessor Management panel appears. Refer to Figure 178 on page 260.

```
CSFCMP00 ----- ICSF Coprocessor Management -----
COMMAND ==>

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                      STATUS
-----
- A06                                               ACTIVE
- A07                                               ACTIVE
- C0          E589C396944007A6 5D40369997A386F4      ONLINE
- C1          0AA379BFD2387960 0367DC04533125FF      ONLINE
- P00         41-00YE1                                ONLINE
- P01         41-00K11                                ONLINE
- P02         41-0A355                                ONLINE
- P03         41-0BA3F                                ONLINE
- P04         41-0RT2T                                DEACTIVATED
- P05         41-00342                                DISABLED
```

Figure 178. Coprocessor Management Panel

On this panel, you can view these options and their values:

Coprocessor

The prefix indicates the type of cryptographic coprocessor or accelerator.

The prefix	Represents a
A	PCI Cryptographic Accelerator
C	Cryptographic Coprocessor Feature
P	PCI Cryptographic Coprocessor

Some servers allow you to partition the processor unit into two sides (side 0 and side 1). The individual central processors, processor storage arrays, and the channel subsystems are associated with side 0 or side 1. The unit on Side 0 is called Coprocessor C0, and the one on Side 1 is called Coprocessor C1.

Module ID/Serial Number

The module ID is the unique 128-bit value that was generated for the CCF during the manufacturing process. The serial number is a number for the PCI Cryptographic Coprocessor.

Status

This field displays the status of the PCICC, the PCICA and the CCF.

State	Indication
ACTIVE (PCICC)	The verification pattern for the SYM-MK matches the verification pattern of the DES master key on the server's Cryptographic Coprocessor Feature. The hash pattern for the ASYM-MK matches the hash pattern of the Signature Master Key (SMK) register on the server's Cryptographic Coprocessor Feature. Requests for services can then be routed to either cryptographic coprocessor.
ACTIVE (PCICA)	The PCICA is available for work.
ACTIVE (CCF)	The DES master key is valid.

ONLINE (PCICC)

The PCI Cryptographic Coprocessor is online, but one or both of the master key verification patterns or hash patterns do not match those of the server's Cryptographic Coprocessor Feature. Requests for services cannot be routed to the PCI Cryptographic Coprocessor.

ONLINE (CCF)

The DES master key is not valid.

OFFLINE (PCICC and PCICA)

A PCICC or PCICA 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 PCICC or PCICA card is configured offline from the Support Element, this status display may not be updated automatically. Users will need to hit enter on this panel to get the latest status.

DISABLED (PCICC and CCF)

The PCI Cryptographic Coprocessor or the Cryptographic Coprocessor Feature has been disabled by the TKE workstation.

DEACTIVATED (PCICC and PCICA)

The PCI Cryptographic Coprocessor or the PCI Cryptographic Accelerator has been deactivated from the Coprocessor Management panel.

TEMP UNAVAILABLE (PCICC and PCICA)

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.

HARDWARE ERROR (PCICC and PCICA)

The reset from a TEMP UNAVAILABLE condition was not successful and the card is unusable.

HARDWARE ERROR (CCF)

A hardware error has been detected.

UNKNOWN: CODE = cccc/ssss (PCICC)

The PCICC has returned an unrecognizable code in response to an attempt to determine its status. The return/reason code appears as the value of CODE.

Displaying coprocessor or accelerator status - PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A

Use the ICSF panels to view the status of the coprocessors. To display coprocessor status:

1. Select option 1, COPROCESSOR MGMT, on the Primary Option panel, as shown in Figure 179 on page 262.

```

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

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 179. Selecting for Coprocessor Status on the Primary Menu Panel

2. The Coprocessor Management panel appears. Refer to Figure 180.

```

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

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

COPROCESSOR  SERIAL NUMBER                      STATUS
-----
- A06                                     ACTIVE
- A07                                     ACTIVE
- X02          42-K0001                     ONLINE
- E04          42-K0043                     DEACTIVATED
- X05          42-K0058                     DISABLED

```

Figure 180. Coprocessor Management Panel

On this panel, you can view these options and their values:

Coprocessor

The prefix indicates the type of cryptographic coprocessor or accelerator.

The prefix	Represents a
A	PCI Cryptographic Accelerator
X	PCIXCC
E	CEX2C
F	CEX2A
G	CEX3C
H	CEX3A

Serial Number

The serial number is a number for the PCIXCC, CEX2C, or CEX3C.

Status

This field displays the status of the PCIXCC, CEX2C, CEX3C, PCICA, CEX2A, or CEX3A.

State	Indication
-------	------------

ACTIVE (PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A)	
---	--

The verification pattern for the SYM-MK matches the verification pattern of the CKDS.

ACTIVE (PCICA, CEX2A, CEX3A)	
-------------------------------------	--

The accelerator is available for work.

ONLINE (PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A)	
---	--

The coprocessor or accelerator is online, but the verification pattern for the SYM-MK does not match the verification pattern of the CKDS.

OFFLINE (PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A)	
--	--

The coprocessor or accelerator 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 card is configured offline from the Support Element, this status display may not be updated automatically. Users will need to hit enter on this panel to get the latest status.

DISABLED (PCIXCC, PCICA, CEX2C, CEX2A, CEX3C, and CEX3A)	
---	--

The coprocessor or accelerator has been disabled by the TKE workstation.

DEACTIVATED (PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A)	
--	--

The coprocessor or accelerator has been deactivated from the Coprocessor Management panel.

TEMP UNAVAILABLE (PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A)	
---	--

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.

HARDWARE ERROR (PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A)	
---	--

The reset from a TEMP UNAVAILABLE condition was not successful and the card is unusable.

UNKNOWN: CODE = cccc/ssss (PCIXCC, CEX2C, CEX3C)	
---	--

The coprocessor has returned an unrecognizable code in response to an attempt to determine its status. The return/reason code appears as the value of CODE.

Changing coprocessor or accelerator status - CCF, PCICC, and PCICA

You can change the status of your PCI cryptographic coprocessors and accelerators, either activating or deactivating them. From the primary menu, select option 1, COPROCESSOR MGMT, and the Coprocessor Management panel is displayed (Figure 181).

```
CSFCMP00 ----- ICSF Coprocessor Management -----
COMMAND ===>

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                      STATUS
-----
D A06                                               ACTIVE
_ A07                                               ACTIVE
_ C0          E589C396944007A6 5D40369997A386F4      ONLINE
_ C1          0AA379BFD2387960 0367DC04533125FF      ONLINE
_ P00         41-00YE1                                ONLINE
_ P01         41-00K11                                ONLINE
_ P02         41-0A355                                ONLINE
_ P03         41-0BA3F                                ONLINE
_ P04         41-0RT2T                                DEACTIVATED
_ P05         41-00342                                DISABLED
```

Figure 181. Coprocessor Management Panel

There are action characters that can be entered on the left of the PCI coprocessor or accelerator number.

Character	Indication
D	Makes a PCICC or PCICA unavailable. The status becomes DEACTIVATED. When the request is made, the status of the PCICC/PCICA may be anything except OFFLINE or DEACTIVATED.
A	Makes available a PCICC or PCICA previously deactivated by a 'D' action character. When the request is made, if the PCICC is online and the master keys are correct, the status will be ACTIVE. If the master keys are incorrect, the status will be ONLINE. When the request is completed successfully, the status of the PCICA is ACTIVE.

Changing coprocessor or accelerator status - PCIXCC, PCICA, CEX2C, CEX3C, CEX2A, and CEX3A

You can change the status of your cryptographic coprocessors and accelerators, either activating or deactivating them. From the primary menu, select option 1, COPROCESSOR MGMT, and the Coprocessor Management panel is displayed (Figure 182 on page 265).

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

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                STATUS
-----
D A06                                     ACTIVE
_ A07                                     ACTIVE
_ X02          42-K0001                         ONLINE
_ E04          42-K0043                         DEACTIVATED
_ X05          42-K0058                         DISABLED
```

Figure 182. Coprocessor Management Panel

Note: This figure is for z890/z990. The coprocessors available for z9 EC and z9 BC are Exx and Fxx. The coprocessors available for z10 EC and z10 BC are Exx, Fxx, Gxx, and Hxx

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 PCIXCC, CEX2C, or CEX3C, 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 ONLINE. For a PCICA, the status will be ACTIVE when the request completes successfully.

Deactivating the last coprocessor

If there are no PCIXCCs, CEX2Cs, or CEX3Cs 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 and online coprocessors. Are you sure you wish to deactivate the last coprocessor?

Press ENTER to confirm the deactivate request.
Press END to cancel the deactivate request.

Figure 183. Coprocessor Management Panel

Displaying coprocessor hardware status - CCF and PCICC

You can use the ICSF panels to view the status of the cryptographic coprocessor key registers, the PCI cryptographic coprocessor, the master key verification patterns, and other information about the cryptographic hardware.

When you enter and activate a DES master key, you change the status of the registers. The cryptographic facility contains several key registers. The master key register contains the active DES master key. For the CCF, the auxiliary key register contains either the old DES master key or a new DES master key prior to it being activated and transferred to the master key register. For the PCICC, there are three registers: one for the old master key, one for the new and one for the current. When you have a PCICC, the old master key is not lost when a new master key is loaded.

In addition, there are also registers for the PKA master keys. When you enter a master key, the Cryptographic Coprocessor Feature or the PCI Cryptographic Coprocessor calculates a verification pattern and a hash pattern for the master key. You can use these patterns to identify master keys.

You can use the panels to display the conditions of the key registers and the verification pattern and hash patterns for the master keys. You may use this information for master key management.

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 -----
COMMAND ==>

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                STATUS
-----
- A06                                     ACTIVE
- A07                                     ACTIVE
S C0      E589C396944007A6 5D40369997A386F4        ACTIVE
C1      0AA379BFD2387960 0367DC04533125FF        ONLINE
S P00      41-00YE1                                ONLINE
- P01      41-00K11                                ONLINE
- P02      41-0A355                                ACTIVE
- P03      41-0BA3F                                ONLINE
- P04      41-0RT2T                                DEACTIVATED
- P05      41-00342                                DISABLED
```

Figure 184. Selecting the coprocessor on the Coprocessor Management Panel

- 2. The Coprocessor Hardware Status panel appears (Figure 185 on page 268). When more than two coprocessors are requested, the status display can be scrolled left and right to show the other coprocessors. You can scroll to the left using PFKey 10 and to the right with PFKey 11.

```
CSFCMP10 ----- ICSF - Coprocessor Hardware Status -----
OPTION ==>
```

CRYPTO DOMAIN: 0

REGISTER STATUS	COPROCESSOR C0	COPROCESSOR P00
		More: +
Crypto Serial Number or Module Id	: E589C39694407A60	41-00YE1
Status	: 5D40C39997A396F0	
	: ACTIVE	ONLINE
DES/Symmetric-Keys Master Key		
New master key register	: FULL	PART FULL
Verification pattern	: 1972BB5791BB2430	2342352352352352
Hash pattern	: 0123456789ABCDEF	A17B93C44D24681A
	: 9691BDA1970BDAA2	806427AAC91221CC
Old master key register	: EMPTY	EMPTY
Verification pattern	:	
Hash pattern	:	
	:	
Current master key register	: VALID	VALID
Verification pattern	: CA6B408A02371B1D	261AAB8A02371705
Hash pattern	: 41DF774FF81547D0	562A5202F8154331
	: 090ABC4539727511	4093990AB1202451
PKA Signature/Asymmetric-Keys Master Key		
New master key register	: N/A	PART FULL
Hash pattern	:	234235236236234D
	:	5678567856785678
Old master key register	: N/A	EMPTY
Hash pattern	:	
	:	
Current master key register	: VALID	VALID
Hash pattern	: 9691BDA1970BDAA2	9691BDA1970BDAA2
	: 1972BB5791BB2430	1972BB5791BB2430
PKA Key Management Master Key register		
Hash pattern	: 123412341241234D	N/A
	: 5678567856785678	
Special Secure Mode	: Enabled	N/A
Environment Control Mask	: FBFEFCF0	N/A
Crypto Configuration Control	: EF569412CD91AB78	N/A
	: 1F25A78BC8ED77A	

Press ENTER to refresh the hardware status display.
Press END to exit to the previous menu.

Figure 185. 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 or Module ID

The serial number is a number for the PCI Cryptographic Coprocessor. The module ID is the unique 128-bit value that was generated for the CCF during the manufacturing process.

Status

This field displays the status of the CCF and the PCICC.

State	Indication
-------	------------

ACTIVE (PCICC)

The verification pattern for the SYM-MK matches the verification pattern of the DES master key on the server's Cryptographic Coprocessor Feature. The hash pattern for the ASYM-MK matches the hash pattern of the Signature Master Key (SMK) register on the server's Cryptographic Coprocessor Feature. Requests for services can then be routed to either cryptographic coprocessor.

ACTIVE (CCF)

The DES master key is valid.

ONLINE (PCICC)

The PCI Cryptographic Coprocessor is online, but one or both of the master key verification patterns or hash patterns do not match those of the server's Cryptographic Coprocessor Feature. Requests for services cannot be routed to the PCI Cryptographic Coprocessor.

ONLINE (CCF)

The DES master key is not valid.

DES/Symmetric-Keys 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 CCF, the new master key is held in an auxiliary key register. This auxiliary key register can contain either a new master key or an old master key. Therefore, a new master key and the old master key cannot coexist.

For the PCICC, 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.

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 and symmetric keys 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.

For the CCF, the old/new master key register is actually the auxiliary master key register. The auxiliary master key register can contain either the new master key or the old master key; therefore a new master key and an old master key cannot coexist at the same time. If an old master key exists, it is lost when you enter a new one.

For the PCICC, 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 DES 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 and symmetric-keys master key register.

State	Indication
EMPTY	You have never entered and set an initial DES/symmetric-keys 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 PKA or 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.

PKA Signature/Asymmetric Master Key

New Master Key register (PCICC only)

This field shows the state of the asymmetric 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 asymmetric master key, or you have just transferred the contents of this register into the asymmetric 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.

Hash Pattern

If the master key register is not EMPTY, a hash pattern is displayed.

Old Master Key register (PCICC only)

This field shows the states of the asymmetric keys old master key register.

State	Indication
EMPTY	You have never changed the asymmetric master key and, therefore, never transferred an asymmetric-keys master key to the asymmetric-keys old master key register. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have changed the asymmetric master key. The asymmetric master key that was current when you changed the master key was placed in the asymmetric old master key register.

Hash Pattern

If the old asymmetric master key register is valid, the panel displays a hash pattern for the asymmetric old master key.

Current Master Key register

This field shows the states of the PKA signature master key and asymmetric master key register.

State	Indication
EMPTY	You have never entered an initial PKA signature master key or an asymmetric 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 PKA signature master key or asymmetric master key on this coprocessor.

Hash Pattern

If the PKA signature master key and asymmetric master key registers are valid, the panel displays a hash pattern for the key. When you enter a new PKA signature master key and asymmetric-keys master key, *record the hash pattern* that appears on the panel. When the PKA signature master key and asymmetric 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 other PCI Cryptographic Coprocessors and one or more Cryptographic Coprocessor Features, the asymmetric master key must be the same on all the PCI cards, and must also be the same as the Signature master key in the Cryptographic Coprocessor Feature. If the status of all these cryptographic coprocessors is valid, the MK hash patterns for each unit should match, because the patterns verify the same key.

Note: An audit trail of the hash patterns that the PCI Cryptographic Coprocessor calculates appears in SMF record type 82.

PKA Key Mangement Master Key register (CCF only)

Hash pattern

You have entered a PKA key management master key and the hash pattern for the key register is shown here.

Special Secure Mode (CCF only)

This field shows if the special secure mode is enabled or disabled. Special secure mode is a lower form of security. This mode allows you to use KGUP to enter clear keys, produce clear PINs, use the secure key import callable

service, and initialize the CKDS. Special secure mode is enabled automatically when you send a KGUP request, provided that the SSM installation option is set to YES.

Environment Control Mask (CCF only)

The environment control mask contains controls for a subset of the components for each domain. This field shows the value of this control.

Note: Selected bits can be changed by the TKE workstation.

Crypto Configuration Control (CCF only)

The crypto configuration control contains controls to enable and disable all the major components of the crypto modules. This field shows the value of this control.

See Appendix A, "CCC Bit Assignments," on page 361 for some selected values.

Note: The CCC cannot be changed.

Displaying coprocessor hardware status - PCIXCC, CEX2C, and CEX3C

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.

When you enter and activate a DES-MK master key, you change the status of the registers. The cryptographic facility contains several key registers. The master key register contains the active DES-MK master key. There are three registers: one for the old master key, one for the new and one for the current. When you have a PCIXCC, CEX2C, or CEX3C, the old master key is not lost when a new master key is loaded.

In addition, there are also registers for the PKA master keys. When you enter a master key, the coprocessor calculates a verification pattern and hash pattern for the DES-MK and a hash pattern for the ASYM-MK. You can use these patterns to identify master keys.

You can use the panels to display the conditions of the key registers and the verification pattern and hash patterns for the master keys. You may use this information for master key management.

To display coprocessor hardware status:

1. From the Coprocessor Management panel, select the coprocessors to be processed by typing an 'S'.

```

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

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

COPROCESSOR  SERIAL NUMBER                                STATUS
-----
-  A06                                               ACTIVE
-  A07                                               ACTIVE
s  X00      42-K0011                                ACTIVE
s  E01      42-K1111                                ONLINE
-  E04      42-K0043                                DEACTIVATED
-  X05      42-K0058                                DISABLED

```

Figure 186. Selecting the coprocessor on the Coprocessor Management Panel

2. The Coprocessor Hardware Status panel appears (Figure 187 on page 275). When more than two coprocessors are requested, the status display can be scrolled left and right to show the other coprocessors. You can scroll to the left using PFKey 10 and to the right with PFKey 11.

```
CSFCMP40 ----- ICSF - Coprocessor Hardware Status -----
OPTION ==>
```

CRYPTO DOMAIN: 8

REGISTER STATUS	COPROCESSOR X00	COPROCESSOR E01
Crypto Serial Number	: 42-K0011	More: + 42-K1111
Status	: ACTIVE	ONLINE
DES Master Key		
New master key register	: EMPTY	EMPTY
Verification pattern	:	
Hash pattern	:	
Old master key register	: EMPTY	VALID
Verification pattern	:	1DD88097BB20C568
Hash pattern	:	F4F5F40AECB1D908
Current master key register	: VALID	VALID
Verification pattern	: CA6B408A02371B1D	261AAB8A02371705
Hash pattern	: 41DF774FF81547D0	562A5202F8154331
	: 090ABC4539727511	4093990AB1202451
AES Master Key		
New master key register	: EMPTY	EMPTY
Verification pattern	:	
Old master key register	: EMPTY	VALID
Verification pattern	:	1DD88097BB20C568
Current master key register	: VALID	VALID
Verification pattern	: CA6B408A02371B1D	261AAB8A02371705
Asymmetric Master Key		
New master key register	: EMPTY	EMPTY
Hash pattern	:	
Old master key register	: EMPTY	EMPTY
Hash pattern	:	
Current master key register	: VALID	VALID
Hash pattern	: 9691BDA1970BDAA2	9691BDA1970BDAA2
	: 1972BB5791BB2430	1972BB5791BB2430

Press ENTER to refresh the hardware status display.
Press END to exit to the previous menu.

Figure 187. 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 PCIXCC, CEX2C, or CEX3C.

I

Status

This field displays the status of the PCIXCC, CEX2C, or CEX3C.

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.

For the PCIXCC, 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.

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.

For the PCIXCC, 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 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.

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 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.

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 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.

Asymmetric Master Key

New Master Key register

This field shows the state of the asymmetric 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 asymmetric master key, or you have just transferred the contents of this register into the asymmetric 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.

Hash Pattern

If the master key register is not EMPTY, a hash pattern is displayed.

Old Master Key register

This field shows the state of the asymmetric old master key register.

State	Indication
EMPTY	You have never changed the asymmetric master key and, therefore, never transferred an asymmetric master key to the asymmetric old master key register. Or you have zeroized the domain from a TKE workstation or the Support Element.
VALID	You have changed the asymmetric master key. The asymmetric master key that was current when you changed the master key was placed in the asymmetric old master key register.

Hash Pattern

If the old asymmetric master key register is valid, the panel displays a hash pattern for the asymmetric old master key.

Current Master Key register

This field shows the states of the asymmetric master key register.

State	Indication
EMPTY	You have never entered an initial asymmetric 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 asymmetric master key on this coprocessor.

Hash Pattern

If the asymmetric master key registers are valid, the panel displays a hash pattern for the key. When you enter a new asymmetric master key, *record the hash pattern* that appears on the panel. When the asymmetric 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.

The asymmetric master key must be the same on all the PCI X cards. If the status of all these cryptographic coprocessors is valid, the MK hash patterns for each unit should match, because the patterns verify the same key.

Note: An audit trail of the hash patterns that the PCIXCC, CEX2C, or CEX3C calculates appears in SMF record type 82.

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 Primary Option panel, as shown in Figure 188 on page 281.

```

CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 3

Enter the number of the desired option.

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

Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 188. Selecting the Installation Options on the Primary Menu Panel

The Installation Options panel appears. Refer to Figure 189.

```

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 189. Installation Options Panel

2. Select option 1, Options, on the Installation Options panel.

The Installation Option Display panel, which is shown in Figure 190 on page 282, appears.

```

CSFSOP10 ----- ICSF - Installation Option Display ROW 1 TO 14 OF 15
COMMAND ==>                                     SCROLL ==> PAGE
Active CKDS: CRYPTOR2.HCRICSF.CKDS
Active PKDS: CRYPTOR2.HCRICSF.PKDS
Active TKDS: CRYPTOR2.HCRICSF.TKDS

OPTION                                     CURRENT VALUE
-----
CHECKAUTH   RACF check authorized callers   YES
COMPAT      Allow CUSP/PCF Compatibility     NO
DOMAIN      Current domain index or usage domain index 0
KEYAUTH     Key Authentication in effect     YES
CKTAUTH     CKT Authentication              NO
SSM         Allow Special Secure Mode       YES
TRACEENTRY  Number of trace entries active   599
USERPARM    User specified parameter data   USERPARM
REASONCODES Source of callable services reason codes ICSF
SYSPLEXCKDS Sysplex consistency for CKDS updates YES,FAIL(YES)
SYSPLEXPKDS Sysplex consistency for PKDS updates NO,FAIL(NO)
SYSPLEXTKDS Sysplex consistency for TKDS updates YES,FAIL(YES)
FIPSMODE    Operate PKCS #11 in FIPS 140-2 mode YES,FAIL(YES)
WAITLIST    Source of CICS Wait List if CICS installed default

***** BOTTOM OF DATA *****

```

Figure 190. Installation Options Display Panel

This panel displays the keyword for each installation option, a brief description, and the current value of the option.

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. 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 the system uses during the startup of ICSF. On the Installation Options Display panel, this data set name is called the active CKDS.

Active PKDS: (data-set-name)

This specifies the name of the PKDS the system uses during the startup of ICSF.

Active TKDS: (data-set-name)

This specifies the name of the TKDS the system uses during the startup of ICSF.

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 class are not performed resulting in a significant performance enhancement for supervisor

state and system key callers. However, the authorization checks are not logged in the RACF SMF records. If you do not specify the CHECKAUTH option, the default is CHECKAUTH(NO).

Value Indication

YES ICSF checks Supervisor State and System Key callers.

NO ICSF does not check Supervisor State and System Key callers, resulting in significant performance enhancement for applications that use ICSF callable services.

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.

DOMAIN(n)

Allows you to access one of several separate sets of master key registers.

Each domain contains these master key registers:

- A master key register that contains the active DES master key
- For the CCF, there is an auxiliary DES master key register that holds either the old or new master key
- If you have a PCICC, there are symmetric master key registers that hold both the old and new master key
- If you have a PCIXCC, CEX2C, or CEX3C, there are symmetric master key registers that hold both the old and new master key
- A PKA key management master key register
- A PKA signature master key register
- If you have a PCICC, there are ASYM-MK registers for the new, old, and current master key.
- If you have a PCIXCC, CEX2C, or CEX3C, there are ASYM-MK registers for the new, old, and current 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 (0 to 15) 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. For more information about logical partitions, see *zSeries PR/SM Planning Guide*.

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.

KEYAUTH(YES, NO or DISABLED)

Indicates whether or not ICSF should authenticate a key entry when it retrieves one from the in-storage cryptographic key data set. If you do not specify the KEYAUTH option, the default value is KEYAUTH(NO).

Value Indication

YES ICSF authenticates the keys. ICSF generates a message authentication code (MAC) for each key entry in the CKDS whenever it creates or updates the key entry. ICSF also performs a MAC verification to ensure that the entry was not changed.

NO ICSF does not authenticate keys retrieved from the in-storage CKDS. ICSF gains a small enhancement of performance.

DISABLED

Record level authentication is disabled in the active CKDS. This option is disabled.

CKTAUTH(YES, NO or DISABLED)

Indicates whether or not ICSF should authenticate each CKDS record when it is read from DASD to create or refresh the in-storage CKDS. If you do not specify the CKTAUTH option, the default value is CKTAUTH(NO).

Value Indication

YES If CKTAUTH(YES) - the MAC authentication code in each record will be authenticated when the record is read from DASD to create or refresh the in-storage CKDS.

NO If CKTAUTH(NO) - MAC authentication is bypassed.

DISABLED

Record level authentication is disabled in the active CKDS. This option is disabled.

SSM(YES or NO)

Indicates whether or not an installation can ever enable special secure mode during the running of ICSF. This mode lowers the security of your system. It allows you to input clear keys by using KGUP, produce clear PINs, use the

Secure Key Import callable service and the initial use of Pass Phrase. SSM(YES) for Pass Phrase is only required for CCF systems. If you do not specify the SSM option, the default value is SSM(NO).

Value Indication

YES Special secure mode is enabled. For z/OS ICSF, SSM(YES) must be specified in order to use KGUP, Secure Key Import callable service, Clear PIN Generate and the initial use of Pass Phrase. SSM(YES) for Pass Phrase is only required for CCF systems.

NO You cannot enable the special secure mode.

TRACEENTRY(n)

Specifies the number, n, of trace buffers to allocate for ICSF tracing. n is a decimal value. The range of valid values is 100 through 10000.

If you do not specify the TRACEENTRY option, the default value is TRACEENTRY(1000).

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.

REASONCODES(ICSF or TSS)

Specifies which set of reason codes the application interface returns.

Value Indication

ICSF ICSF reason codes are returned.

TSS TSS reason codes are returned.

ICSF is the default.

SYSPLEXCKDS(YES or NO,FAIL(YES or NO))

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 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 CKDS by other members of the ICSF sysplex group. The default is SYSPLEXCKDS(NO,FAIL(NO)).

SYSPLXPKDS(YES or NO,FAIL(YES or NO))

Displays the current value of the SYSPLXPKDS option. The values of the option can be YES or NO, with the default being NO. If SYSPLXPKDS(NO,FAIL(fail-option)) is specified, no XCF signalling will be performed when an update to a PKDS record occurs. If SYSPLXPKDS(YES,FAIL(fail-option)) is specified, the support described in "PKDS management in a sysplex" on page 183 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(YES or NO))

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 189 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)).

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.

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*.

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 293. For a description of how to display installation-defined callable service information, see “Displaying installation-defined callable services” on page 301.

Displaying PCICC default roles

Use the ICSF panels to display the default role for the coprocessor. All the access control points enabled will be listed.

1. Select option 1, COPROCESSOR MGMT, on the Primary Option panel, as shown in Figure 191.

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

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 191. Selecting for Coprocessor Status on the Primary Menu Panel

2. The Coprocessor Management panel appears. Refer to Figure 192.

```
CSFCMP00 ----- ICSF Coprocessor Management -----
COMMAND ==>

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

COPROCESSOR  MODULE ID/SERIAL NUMBER                STATUS
-----
- A06                                     ACTIVE
- A07                                     ACTIVE
- C0      E589C396944007A6 5D40369997A386F4         ACTIVE
- C1      0AA379BFD2387960 0367DC04533125FF         ACTIVE
- P00      41-00YE1                                     ONLINE
R P01      41-00K11                                     ACTIVE
- P02      41-0A355                                     ACTIVE
- P03      41-0BA3F                                     ONLINE
- P04      41-0RT2T                                     DEACTIVATED
- P05      41-00342                                     DISABLED
```

Figure 192. Coprocessor Management Panel

3. Select the PCICC by entering an 'R' to the left of the coprocessor. Press enter and the Status Display panel appears (Figure 193 on page 288).

Note: The default role can be changed with a TKE workstation. See *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

Access Control Manager - Read role
Authorize UDX
Clear New ASYM Master Key Register
Clear New DES Master Key Register
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
Combine ASYM Master Key Parts
Combine DES Master Key Parts
Control Vector Translate
Cryptographic Variable Encipher
Data Key Export
Data Key Export - Unrestricted
Data Key Import
Data Key Import - Unrestricted
Digital Signature Generate
Diversified Key Generate - single length or same halves
Diversified Key Generate - CLR8-ENC
Diversified Key Generate - SESS-XOR
Diversified Key Generate - TDES-DEC
Diversified Key Generate - TDES-ENC
DATAM Key Management Control
DES Key Token Change
Encrypted PIN Generate - GBP
Encrypted PIN Generate - Interbank
Encrypted PIN Generate - 3624
Encrypted PIN Translate - Reformat
Encrypted PIN Translate - Translate
Encrypted PIN Verify - GBP
Encrypted PIN Verify - Interbank
Encrypted PIN Verify - VISA PVV
Encrypted PIN Verify - 3624
Generate CVV
Key Export
Key Export - Unrestricted
Key Generate - OP,IM,EX
Key Generate - OPIM,OPEX,IMEX,etc.
Key Generate - OPIM,OPEX,IMEX,etc. extended
Key Generate - SINGLE-R
Key Import
Key Import - Unrestricted
Key Part Import - first key part
Key Part Import - middle and last
Key Part Import - Unrestricted
Key Test
Key Translate

Figure 193. Default Role Status Display Panel

```

Load First ASYM Master Key Part
Load First DES Master Key Part
MAC Generate
MAC Verify
Prohibit Export
PKA Decrypt
PKA Encrypt
PKA Key Generate
PKA Key Generate - Clear
PKA Key Generate - Clone
PKA Key Import
PKA Key Token Change RTCMK
Reencipher CKDS
Retained Key Delete
Retained Key List
Secure Key Import - DES,IM
Secure Key Import - DES,OP
Secure Messaging for Keys
Secure Messaging for PINs
Set ASYM Master Key
Set DES Master Key
Symmetric Key Export - DES, PKCS-1.2
Symmetric Key Export - DES, ZERO-PAD
Symmetric Key Generate - DES, PKA92
Symmetric Key Generate - DES, PKCS-1.2
Symmetric Key Generate - DES, ZERO-PAD
Symmetric Key Import - DES, PKA92 KEK
Symmetric Key Import - DES, PKCS-1.2
Symmetric Key Import - DES, ZERO-PAD
SET Block Compose
SET Block Decompose
SET Block Decompose - PIN Extension IPINENC
SET Block Decompose - PIN Extension OPINENC
UKPT - PIN Verify, PIN Translate
Verify CVV

```

Figure 194. Default Role Status Display Panel – part 2

Displaying PCIXCC, CEX2C, and CEX3C default roles

Use the ICSF panels to display the default role for the coprocessor. All the access control points enabled will be listed.

1. Select option 1, COPROCESSOR MGMT, on the Primary Option panel, as shown in Figure 195 on page 290.

```

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

Enter the number of the desired option.

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

    Licensed Materials - Property of IBM

    5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
    US Government Users Restricted Rights - Use, duplication or
    disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 195. Selecting for Coprocessor Status on the Primary Menu Panel

2. The Coprocessor Management panel appears. Refer to Figure 196.

```

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

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

COPROCESSOR  SERIAL NUMBER                      STATUS
-----
- A06                                     ACTIVE
- A07                                     ACTIVE
R X02         42-0A355                        ONLINE
- X03         42-0BA3F                        ONLINE
- X04         42-0RT2T                        DEACTIVATED
- X05         42-00342                        DISABLED

```

Figure 196. Coprocessor Management Panel

3. Select the PCIXCC, CEX2C, or CEX3C by entering an 'R' to the left of the coprocessor. Press enter and the Status Display panel appears (Figure 197 on page 291).

Note: A TKE is required in order to change the default role. See *z/OS Cryptographic Services ICSF TKE PCIX Workstation User's Guide*.

```

CSFCMP30 ----- ICSF Status Display -----
COMMAND ==>

Enabled access control points from the default role for X02, domain 0.

Access Control Manager - Read role
Authorize UDX
Clear Key Import/Multiple Clear Key Import - DES
Clear New AES Master Key
Clear New ASYM Master Key Register
Clear New DES Master Key Register
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
Combine AES Master Key Parts
Combine ASYM Master Key Parts
Combine DES Master Key Parts
Control Vector Translate
Cryptographic Variable Encipher
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 - single length or same halves
Diversified Key Generate - CLR8-ENC
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 Key Token Change
Encipher - DES
Encrypted PIN Generate - GBP
Encrypted PIN Generate - Interbank
Encrypted PIN Generate - 3624
Encrypted PIN Translate - Reformat
Encrypted PIN Translate - Translate
Encrypted PIN Verify - GBP
Encrypted PIN Verify - Interbank
Encrypted PIN Verify - VISA PVV
Encrypted PIN Verify - 3624
Generate CVV
Key Export
Key Export - Unrestricted
Key Generate - OP,IM,EX
Key Generate - OPIM,OPEX,IMEX,etc.
Key Generate - OPIM,OPEX,IMEX,etc. extended
Key Generate - SINGLE-R

```

Figure 197. Default Role Status Displayed for a system without TKE connected

```

CSFCMP30 ----- ICSF Status Display -----
COMMAND ==>

Key Import
Key Import - Unrestricted
Key Part Import - first key part
Key Part Import - middle and last
Key Part Import - ADD-PART
Key Part Import - COMPLETE
Key Part Import - RETRKPR
Key Part Import - Unrestricted
Key Test
Key Translate
Load First AES Master Key Part
Load First ASYM Master Key Part
Load First DES Master Key Part
Multiple Clear Key Import/Multiple Secure Key Import - AES
MAC Generate
MAC Verify
NOCV KEK usage for export-related functions
NOCV KEK usage for import-related functions
Prohibit Export
Prohibit Export Extended
PCF CKDS conversion utility
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
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
Reencipher CKDS
Remote Key Export - Gen or export a non-CCA node Key
Retained Key Delete
Retained Key List
Secure Key Import - DES,IM
Secure Key Import - DES,OP
Secure Messaging for Keys
Secure Messaging for PINs
Set AES Master Key
Set ASYM Master Key
Set DES Master Key
Symmetric Algorithm Decipher - secure AES keys
Symmetric Algorithm Encipher - secure AES keys
Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2
Symmetric Key Export - AES, ZERO-PAD
Symmetric Key Export - DES, PKCS-1.2
Symmetric Key Export - DES, ZERO-PAD

```

Figure 198. Default Role Status Displayed for a system without TKE connected - part 2

```

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 - 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
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
UKPT - PIN Verify, PIN Translate
Verify CVV

```

Figure 199. Default Role Status Displayed for a system without TKE connected – part 3

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 Primary Option panel, as shown in Figure 200.

```
CSF@PRIM ---- Integrated Cryptographic Service Facility -----
OPTION ==> 3

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 200. Selecting the Installation Options and Hardware Status Option on the Primary Menu Panel

The Installation Options panel appears. Refer to Figure 201.

```
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
```

Figure 201. Installation Options Panel

2. Select option 2, Exits, on the Installation Options panel.
The first of the Installation Exits Display panels appears. Refer to Figure 202 on page 295.


```
CSFSOP30 ----- ICSF - Installation Exits Display ---- ROW 1 TO 18 OF 70
COMMAND ==>
```

ICSF NAME	LOAD MODULE	OPTIONS
-----	-----	-----
CSFAEGN		*** No Exit Name was specified ***
CSFAKEX		*** No Exit Name was specified ***
CSFAKIM		*** No Exit Name was specified ***
CSFAKTR		*** No Exit Name was specified ***
CSFATKN		*** No Exit Name was specified ***
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 ***
CSFCTT		*** No Exit Name was specified ***
CSFCTT1		*** 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

Figure 202. First Installation Exits Display Panel

The Installation Exits Display panel 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 second panel of exits is shown in Figure 203 on page 296.

CSFSOP30 ----- ICSF - Installation Exits Display --- ROW 19 TO 36 OF 70
 COMMAND ==>

ICSF NAME	LOAD MODULE	OPTIONS
-----	-----	-----
CSFEMK		*** No Exit Name was specified ***
CSFENC		*** No Exit Name was specified ***
CSFENC1		*** No Exit Name was specified ***
CSFEPG		*** No Exit Name was specified ***
CSFESECI		*** No Exit Name was specified ***
CSFESECK		*** No Exit Name was specified ***
CSFESECS		*** No Exit Name was specified ***
CSFESECT		*** No Exit Name was specified ***
CSFEXIT2		*** No Exit Name was specified ***
CSFEXIT3		*** No Exit Name was specified ***
CSFEXIT4		*** No Exit Name was specified ***
CSFEXIT5		*** No Exit Name was specified ***
CSFGKC		*** No Exit Name was specified ***
CSFKEX		*** No Exit Name was specified ***
CSFKGN		*** No Exit Name was specified ***
CSFKGUP		*** No Exit Name was specified ***
CSFKIM		*** No Exit Name was specified ***
CSFKPI		*** No Exit Name was specified ***
CSFKRC		*** No Exit Name was specified ***
CSFKRD		*** No Exit Name was specified ***
CSFKRR		*** No Exit Name was specified ***
CSFKRW		*** No Exit Name was specified ***
CSFKTC		*** No Exit Name was specified ***
CSFKTR		*** No Exit Name was specified ***
CSFKYT		*** No Exit Name was specified ***
CSFKYTX		*** No Exit Name was specified ***
CSFMDG		*** No Exit Name was specified ***

Figure 203. Second Installation Exits Display Panel

The third panel of exits is shown in Figure 204 on page 297.

```
CSFSOP30 ----- ICSF - Installation Exits Display ---- ROW 37 TO 54 OF 70
COMMAND ==>
```

ICSF NAME	LOAD MODULE	OPTIONS
CSFMDG1		*** No Exit Name was specified ***
CSFMGN		*** No Exit Name was specified ***
CSFMGN1		*** No Exit Name was specified ***
CSFMVR		*** No Exit Name was specified ***
CSFMVR1		*** No Exit Name was specified ***
CSFOWH		*** No Exit Name was specified ***
CSFOWH1		*** No Exit Name was specified ***
CSFPCI		*** No Exit Name was specified ***
CSFPCU		*** No Exit Name was specified ***
CSFPEX		*** No Exit Name was specified ***
CSFPEXX		*** No Exit Name was specified ***
CSFPGN		*** No Exit Name was specified ***
CSFPKD		*** No Exit Name was specified ***
CSFPKE		*** No Exit Name was specified ***
CSFPKG		*** No Exit Name was specified ***
CSFPKI		*** No Exit Name was specified ***
CSFPKRC		*** No Exit Name was specified ***
CSFPKRD		*** No Exit Name was specified ***
CSFPKRR		*** No Exit Name was specified ***
CSFPKRW		*** No Exit Name was specified ***
CSFPKSC		*** No Exit Name was specified ***
CSFPKTC		*** No Exit Name was specified ***
CSFPKX		*** No Exit Name was specified ***
CSFPTR		*** No Exit Name was specified ***
CSFPVR		*** No Exit Name was specified ***
CSFRKD		*** No Exit Name was specified ***
CSFRKL		*** No Exit Name was specified ***
CSFRNG	EXITRNG	EXIT - Do not call this exit again, if it fails
CSFRNGL		*** No Exit Name was specified ***
CSFRSWS		*** No Exit Name was specified ***
CSFRTC		*** No Exit Name was specified ***
CSFSBC		*** No Exit Name was specified ***

Figure 204. Third Installation Exits Display Panel

The fourth panel of exits is shown in Figure 205 on page 298.

```
CSFSOP30 ----- ICSF - Installation Exits Display ----- ROW 55 TO 70 OF 70
COMMAND ===>
```

ICSF NAME	LOAD MODULE	OPTIONS
CSFSBD		*** No Exit Name was specified ***
CSFSKI		*** No Exit Name was specified ***
CSFSKM		*** No Exit Name was specified ***
CSFSKY		*** No Exit Name was specified ***
CSFSPN		*** No Exit Name was specified ***
CSFSRRW		*** No Exit Name was specified ***
CSFSSWS		*** No Exit Name was specified ***
CSFSYG		*** No Exit Name was specified ***
CSFSYI		*** No Exit Name was specified ***
CSFSYX		*** No Exit Name was specified ***
CSFTBC		*** No Exit Name was specified ***
CSFTCK		*** No Exit Name was specified ***
CSFTRV		*** No Exit Name was specified ***
CSFUDK		*** No Exit Name was specified ***
*****BOTTOM OF DATA*****		

Figure 205. Fourth Installation Exits Display Panel

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 17 shows the names for some general ICSF exits. Table 18 and Table 19 on page 301 show the ICSF name for each callable service exit.

Table 17. 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 18. Callable Service and its Exit Identifier

Service	Exit Identifier
ANSI X9.17 EDC generate	CSFAEGN
ANSI X9.17 Key Export	CSFAKEX
ANSI X9.17 Key Import	CSFAKIM
ANSI X9.17 Key Translate	CSFAKTR
ANSI X9.17 Transport Key Partial Notarize	CSFATKN

Table 18. Callable Service and its Exit Identifier (continued)

Service	Exit Identifier
Clear PIN Encrypt	CSFCPE
Clear PIN Generate Alternate	CSFCPA
Clear Key Import	CSFCKI
Cipher/Decipher	CSFEDC
Cipher Text Translate	CSFCTT
Cipher Text Translate (with ALET)	CSFCTT1
Control Vector Translate	CSFCVT
Cryptographic Variable Encipher	CSFCVE
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
Encode	CSFECO
Encipher under Master Key	CSFEMK
Encipher	CSFENC
Encipher (with ALET)	CSFENC1
Encrypted PIN Generate	CSFEPPG
Key Export	CSFKEX
Key Generate	CSFKGN
Key Import	CSFKIM
Key Part Import	CSFKPI
Key Record Create	CSFKRC
Key Record Delete	CSFKRD
Key Record Read	CSFKRR
Key Record Write	CSFKRW
Key Test	CSFKYT
Key Test Extended	CSFKYTX
Key Translate	CSFKTR
MAC Generate	CSFMGN
MAC Generate (with ALET)	CSFMGN1
MAC Verify	CSFMVR
MAC Verify (with ALET)	CSFMVR1
MDC Generate	CSFMDG
MDC Generate (with ALET)	CSFMDG1
Multiple Clear Key Import	CSFCKM
Multiple Secure Key Import	CSFSCKM

Table 18. Callable Service and its Exit Identifier (continued)

Service	Exit Identifier
One-Way Hash Generate	CSFOWH
One-Way Hash Generate (with ALET)	CSFOWH1
PCI Interface	CSFPCI
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 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
Retained Key Delete	CSFRKD
Retained Key List	CSFRKL
Secure Key Import	CSFSKI
Secure Messaging for Keys	CSFSKY
Secure Messaging for PINs	CSFSPN
SET Block Compose	CSFSBC
SET Block Decompose	CSFSBD
Symmetric Algorithm Decipher	CSFSAD
Symmetric Algorithm Encipher	CSFSAE
Symmetric Key Generate	CSFSYG
Symmetric Key Import	CSFSYI
Symmetric Key Export	CSFSYX
Symmetric MAC Generate	CSFSMG
Symmetric MAC Generate (with ALET)	CSFSMG1
Symmetric MAC Verify	CSFSMV
Symmetric MAC Verify (with ALET)	CSFSMV1
Transaction Validation	CSFTRV

Table 18. Callable Service and its Exit Identifier (continued)

Service	Exit Identifier
Transform CDMF Key	CSFTCK
Trusted Block Create	CSFTBC
User Derived Key	CSFUDK
VISA CVV Service Generate	CSFCSG
VISA VISA CVV Service Verify	CSFCSV

Table 19. 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.

- 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:

- Select option 3, OPSTAT, on the Primary Option panel, as shown in Figure 206 on page 302.

```

CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 3

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

      5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

```

Figure 206. Selecting the Installation Options and Hardware Status Option on the Primary Menu Panel

The Installation Options panel appears. Refer to Figure 207.

```

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 207. Installation Options Panel

2. Select option 3, Services, on the Installation Options Status panel.

The Installation Defined Services panel appears. Refer to Figure 208.

```

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 208. 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 identifies 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 12. Managing User Defined Extensions

User Defined Extensions (UDX) support allows you to request implementation of a customized cryptographic callable service. This support is available for the PCICC, PCIXCC, CEX2C, and CEX3C. 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.

With a special contract with IBM, you can develop and load your own UDXs for the z990, z890, z9 EC, z9 BC, z10 EC and z10 BC.

Note: A TKE Workstation is required to enable the access control points for UDXs.

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, PCIXCC, CEX2C, or CEX3C
- Display a list of all PCICCs, PCIXCCs, CEX2Cs, or CEX3Cs on which a specific UDX is authorized
- Authorize a UDX on any PCI cryptographic coprocessor in the system

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----
OPTION ==> 9

Enter the number of the desired option.

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

      Licensed Materials - Property of IBM

5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

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

Figure 209. Selecting the UDX MGMT Option on the ICSF Primary Menu Panel

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
- 3 Authorize a UDX

Figure 210. User Defined Extensions Management Panel

Display UDXs for a coprocessor

A panel similar to Figure 211 is displayed when option 1 is selected. If you are running on a IBM @server zSeries 990 or IBM @server zSeries z890, you will see a list of PCIxCCs/CEX2Cs. For a z9 EC or z9 BC, you will see a list of CEX2Cs. For the z10 EC and z10 BC you will see a list of CEX2Cs and CEX3Cs.

```
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
-----	-----	-----
P00	41-00YE1	ACTIVE
P01	41-00K11	ACTIVE
P02	41-0A355	ACTIVE
P03	41-0BA3F	ACTIVE
P04	41-0RT2T	ACTIVE
P07	41-00B4M	ACTIVE

Figure 211. 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 212 on page 307 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 212. 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 213. Coprocessors for Authorized UDXs Panel

Use this panel to specify the User Defined Extension id to be queried. A panel similar to Figure 214 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
  -----
  P00              41-00YE1            ACTIVE
  P01              41-00K11            ACTIVE
  P04              41-0RT2T            ACTIVE
*****Bottom of data*****

```

Figure 214. Coprocessors for Authorized UDXs Panel

Authorize a UDX

A panel similar to Figure 215 is displayed when option 3 is selected from the User Defined Extensions Management Panel. If you are running on a z890 or z990, you will see a list of PCIXCCs/CEX2Cs. If you are running on a z9 EC or z9 BC, you will see a list of CEX2Cs. If you are running on a z10 EC and z10 BC, you will see a list of CEX2Cs and CEX3Cs.

```
CSFUDX50 ----- ICSF - Authorize User Defined Extension -- Row 1 to 1 of 6
COMMAND ==>                                           SCROLL==> PAGE

UDX id ==>
Password==>

Select the coprocessors to be processed and press ENTER.

COPROCESSOR      SERIAL NUMBER      STATUS
-----
P00              41-00YE1             ACTIVE
P01              41-00K11             ACTIVE
P02              41-0A355             ACTIVE
P03              41-0BA3F             ONLINE
P04              41-0RT2T             ACTIVE
P07              41-00B4M             ACTIVE
*****Bottom of data*****
```

Figure 215. Authorize UDXs Panel

If your UDX was developed for your installation by IBM Global Services, you may have been provided a password associated with the UDX.

Use this panel to authorize a specific User Defined Extension on one or more PCI Cryptographic Coprocessors. (UDXs on PCIXCCs, CEX2Cs, and CEX3Cs do not require authorization via this panel.)

Enter the the two character id in the UDX id field. Enter the sixteen hexadecimal characters of the password in the Password field. Use an **s** to select the coprocessors where the UDX will be authorized.

Chapter 13. 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 Primary Option panel, and press ENTER. Refer to Figure 216.

```
CSF@PRIM ----- Integrated Cryptographic Service Facility -----  
OPTION ==> 5  
  
Enter the number of the desired option.  
  
 1 COPROCESSOR MGMT - Management of Cryptographic Coprocessors  
 2 MASTER KEY MGMT  - Master key set or change, CKDS/PKDS processing  
 3 OPSTAT           - Installation options  
 4 ADMINCNTL        - Administrative Control Functions  
 5 UTILITY           - ICSF Utilities  
 6 PPINIT           - Pass Phrase Master Key/CKDS Initialization  
 7 TKE              - TKE Master and Operational key processing  
 8 KGUP             - Key Generator Utility processes  
 9 UDX MGMT         - Management of User Defined Extensions  
  
        Licensed Materials - Property of IBM  
  
        5694-A01 (C) Copyright IBM Corp. 1990, 2009. All rights reserved.  
        US Government Users Restricted Rights - Use, duplication or  
        disclosure restricted by GSA ADP Schedule Contract with IBM Corp.  
  
Press ENTER to go to the selected option.  
Press END   to exit to the previous menu.
```

Figure 216. Selecting the Utilities Option on the Primary Menu Panel

The Utilities panel appears. See Figure 217 on page 310.

```

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 217. Selecting the Encode Option on the Utilities Panel

2. Select option 1, Encode, on this panel.
The Encode panel appears. See Figure 218.

```

CSFEC000 ----- 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 218. 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 219 on page 311.


```
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 219. Selecting the Decode Option on the Utilities Panel

2. Select option 2, Decode, on this panel.
The Decode panel appears. See Figure 220.

```
CSFEC000 ----- 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 220. 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 14. Using the Utility Panels to Manage Keys in the PKDS

This capability enhances the ICSF utilities panel, option 6 PKDSKEYS, to provide PKDS key management capability. This new function gives customers the ability to:

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

These functions are intended for use with the Encryption Facility, but may be used for other purposes.

To use the full function of the ICSF PKDS Key Management panels, you must have a PCICC, PCIXCC, CEX2C, or CEX3C cryptographic coprocessor. If you do not have one of these coprocessors, you cannot generate key pairs using the panels.

RACF Protecting ICSF Services used by the New Panels

ICSF uses these ICSF callable services to create or delete PKDS records and export or import RSA keys to X.509 certificates:

CSNDKRR

Ensures that the specified PKDS label does not already 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*.

Follow these steps to manage keys in the PKDS.

Select option 6, PKDSKEYS, on the ICSF Utilities panel as shown in Figure 221.

```
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 221. Selecting the PKDSKEYS option on the ICSF Utilities Panel

If option 6 is selected on the utilities panel, the ICSF - PKDS Keys is presented:

```
CSFPKY00 ----- ICSF - PKDS Keys -----
COMMAND ==>

Enter the PKDS record's label for the actions below
==>

Select one of the following actions then press ENTER to process:

- Generate a new PKDS key pair record
  Enter the key length ==>      512, 1024, 2048, 3072 or 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 222. ICSF PKDS Keys Panel

From this panel you can manage 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 public/private PKDS key pair record
- Delete an existing key record
- Export a public key to an X.509 certificate for importation elsewhere
- Import a public key from an X.509 certificate received from elsewhere

Generate a new RSA public/private PKDS key pair record

The key pair generated may be used to encrypt and recover archive data. It may also be used to recover encrypted data transmitted to you by another party.

- The key length in bits must be specified (512, 1024, 2048, 3072 or 4096).
- The private key name may also be specified, but is optional.
- Blank is the default for the private key name.
- Callable services:
 - CSNDKRR - ensures that the specified PKDS label doesn't already exist
 - CSNDPKB - builds the skeleton key token
 - CSNDPKG - generates the key pair
 - CSNDKRC - creates the PKDS record

Note:

1. The key pair created may be used for generating and verifying digital signatures and key management.
2. The public exponent used for all keys generated through this service is X'010001'.

Delete an existing key record

This service may be used to delete any PKDS record, whether or not created by this utility.

- If Delete is selected, a new popup panel Delete PKDS Key Confirmation (CSFPKY0P) is displayed forcing the user 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 will no longer be 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 may be sent to another party. Then you may receive data from another party enciphered under the public key which you may recover using the same PKDS record.

- The certificate created will be stored in an MVS physical sequential data set.
- The output data set will be 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 will be destroyed and the data set reallocated new.
- The data set can not be a PDS or PDS member.
- You may specify a value for the subject's common name in the certificate, if desired.
 - If no value is specified, the PKDS record's label will be used as the common name.
- Callable services:
 - CSNDKRR - reads the record from the PKDS

- CSNDPKX - extracts just the public key from the record
- CSNBOWH - hashes the to-be-signed portion of the generated certificate
- CSNDDSG - signs the hash

Note:

1. The key record specified must be a public or private key pair record and must support signing.
2. The certificate's validity date range is hard coded to be July 1, 2005 - December 31, 2040 UTC.
3. The certificate created will be 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. Once complete, you may send the other party data enciphered under the public which the other party can recover.

- The data set name 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 can not 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 223. 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 224. 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 225. PKDS Public Key Import Successful

Failure

For the various functions, these expected errors will 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 already exists (Generate and Import only)
6. PKDS label not found (Delete and Export only)

7. Specifying a PDS member (Import and Export only)
8. Can't 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 226. 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 227. 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 228. PKDS Public Key Import Failure

Chapter 15. 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 20 on page 322 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 20. Token access levels

CRYPTOZ resource	SAF access level		
	READ	UPDATE	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

Notes:

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 21 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 21.

Table 21. 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

Follow these steps to use the PKCS11 token browser panel utility.

Select option 7, PKCS11 TOKEN, on the ICSF Utilities panel as shown in Figure 229.

```
CSFUTL00 ----- ICSF - Utilities -----
OPTION ==> 7

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 ==>
```

Figure 229. Selecting the PKCS11 TOKEN option on the ICSF Utilities Panel

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 230. 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 231. 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 232. 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 233. 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 234. 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 235. 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 1      DATA      PRIVATE: TRUE      MODIFIABLE: TRUE
  LABEL:       Data for lastpass
  APPLICATION: 90893E31
  OBJECT ID:   Not-specified
  VALUE:       0123456789ABCDEF

_ Object 2      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 3      ???       PRIVATE: ???       MODIFIABLE: ???
  NOT AUTHORIZED TO BROWSE

_ Object 4      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 5      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 6      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 7:     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 236. 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 1      from token label: SAMPLE.TOKEN
```

```
Select an Action:
```

- 1 Modify one or more fields with the new values specified
- 2 Delete the entire object

```
-----
OBJECT CLASS:      DATA                               More:  +
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 237. 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 2 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:          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
34310B3009060355040613025553310C | 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
533131205465737420456E642D456E74 | S11 Test End-Ent
69747930819F300D06092A864886F70D | ity0..0...*.H...
010101050003818D0030818902818100 | .....0.....
AAA38A1F45C93C1772C5AC223A1DAE32 | ...E.<.r..":..2
F932C5347931CFF6696D9A4205A5957D | .2.4y1..im.B...}
8CD83CEF0719E82DDEF5E4C5FB53E89D | ..<....-.....S..
80927186B89A619756CF75500CCD5C47 | ..q...a.V.uP..G
9F46E01C7EAD0061ABF8CB2357C9603 | .F.v.....5|..
3CC1E7E464BF4289AE0AD51E9FA2E86C | <...d.B.....l
C80504552C2E35C0F5BE4F13ACEC8253 | ...U,.5...0....S
```

Press ENTER to process.

Press END to exit to the previous menu.

COMMAND ==>

Figure 238. 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:

CSFTBR33 ----- ICSF Token Management - Secret Key Object Details -----

Object 4 from token label: SAMPLE.TOKEN

Select an Action:

- 1 Modify one or more fields with the new values specified
- 2 Delete the entire object

OBJECT CLASS:	SECRET KEY	More:	+
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 to TRUE)	
	New value:	FALSE	
SENSITIVE:	FALSE	(Cannot be changed from TRUE to FALSE)	
	New value:	TRUE	
ALWAYS SENSITIVE:	FALSE		
NEVER EXTRACTABLE:	FALSE		
VALUE:	NOT DISPLAYABLE		
VALUE LEN:	16		
FIPS140	FALSE		
APPLICATION:	90893E31		

Press ENTER to process.

Press END to exit to the previous menu.

COMMAND ==>

Figure 239. ICSF Token Management - Secret Key Object Details panel

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:

CSFTBR31 ----- ICSF Token Management - Public Key Object Details -----

Object 5 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:      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:        20050103
New value:         YYYYMMDD
END DATE:          20071231
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
APPLICATION:        90893E31
MODULUS BITS:       4096
PUBLIC EXPONENT:    010001
MODULUS:
F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D
18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC
662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61
5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805
CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E
69AD03FBFBBC5990EDDF8C1A34D607AC3B7728D7E6ABBD566A626980E0D888
C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B
95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562
3EACCC1C25742C25924612B407869788F3236AF037B7D7EBB03C0FB6529A376
CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF
26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73
C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0
A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD
7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0
10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553
1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753
COMMAND ==>
```

Figure 240. ICSF Token Management - Public Key Object Details panel

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, or Elliptic Curve) selected.

Table 22. Information displayed in Public Key Object Details panel for RSA, DSA, Diffie-Hellman, and Elliptic Curve 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 69AD03FBFBBC5990EDDF8C1A34D607AC3B7728D7E6ABDA566A626980E0D888 C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B 95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562 3EACCC1C25742C25924612B407869788F3236AF037B7D7EBB03C0FB6529A376 CF8161AACA0B9C3D285D772C71B78264B56DE152B8B70975CE8857D3EB048FF 26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73 C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0 A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDA6A366AD 7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0 10B582BB67A9048FEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553 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 1362CDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1 1D4CD1CF810767BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40 VALUE: 3992F874061239B0A0B2E52BDBC33237A1CEAB624B613AD91D23CDDCCF58D575 1CB5FE0364D37BF74721AA5473DD1ECC2E65B82138BE7103477C438B3486C548 0CCAD36D7882C9659CA32744E776DE894193953F6DF32C8AC3ACFC0A364A641A A19B74BDC43E6EC84D8CB409B46A82D666A7F963D31A2CC897B971D378959509

Table 22. Information displayed in Public Key Object Details panel for RSA, DSA, Diffie-Hellman, and Elliptic Curve 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 69AD03FBFBCC5990EDDF8C1A34D607AC3B7728D7E6ABDA566A626980E0D888 C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B 95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562 BASE: 3EACCC1C25742C25924612B407869788F3236AF037B7D7EBBB03C0FB6529A376 CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8857D3EB048FF 26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73 C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0 A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD 7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0 10B582BB67A9048FEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553 1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753 VALUE: 2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33 C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9 4490C4837B5656776E9FFDA073EAED869B19F7E197970DBE5665E8F87F964C57 F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE 1362CDD0387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1 1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40
Elliptic Curve	EC	The elliptic curve parameters and the elliptic curve point. For example: – EC PARAMS*: Named Curve – secp521r1 – EC POINT*: 3992F874061239B0A0B2E52BDBC33237A1CEAB624B613AD91D23CDDCCFC58D575 1CB5FE0364D37BF74721AA5473DD1ECC2E65B82138BE7103477C438B3486C548 0CCAD36D7882C9659CA32744E776DE894193953F6DF32C8AC3ACFC0A364A641A A19B74BDC43E6EC84D8CB409B46A82D666A7F963D31A2CC897B971D378959509 308189028181008B53

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:

CSFTBR32 ----- 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

OBJECT CLASS:	PRIVATE KEY	More:	+
PRIVATE:	TRUE		
MODIFIABLE:	TRUE		
LABEL:	privatekey cx021A		
SUBJECT*:	New value: Not-specified		
ID:	83A7F0F2F1C1		
KEY TYPE:	New value: RSA		
START DATE:	20050103	YYYYMMDD	
END DATE:	New value: 20071231	YYYYMMDD	
DERIVE:	FALSE		
LOCAL:	FALSE		
KEY GEN MECHANISM:	UNAVAILABLE INFORMATION		
DECRYPT:	FALSE		
SIGN:	New value: TRUE		
SIGN RECOVER:	New value: FALSE		
UNWRAP:	New value: TRUE		
EXTRACTABLE:	New value: FALSE		
SENSITIVE:	New value: TRUE		

Figure 241. ICSF Token Management - Private Key Object Details panel – Part 1

```

ALWAYS SENSITIVE:          FALSE
NEVER EXTRACTABLE:        FALSE
FIPS140                  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
69AD03FBFBCC5990EDDF8C1A34D607AC3B7728D7E6ABBD4566A626980E0D888
C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B
95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562
3EACCC1C25742C25924612B407869788F3236AF037B7D7EBB03C0FB6529A376
CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF
26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73
C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0
A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDAA6A366AD
7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0
10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553
1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753
Press ENTER to process.
Press END to exit to the previous menu.

COMMAND ==>

```

Figure 242. ICSF Token Management - Private Key Object Details panel – Part 2

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, or Elliptic Curve) selected.

Table 23. Information displayed in Private Key Object Details panel for RSA, DSA, Diffie-Hellman, and Elliptic Curve 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: <pre> 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 69AD03FBFBCC5990EDDF8C1A34D607AC3B7728D7E6ABBD566A626980E0D888 C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B 95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562 3EACCC1C25742C25924612B407869788F3236AF037B7D7EBB03C0FB6529A376 CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF 26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73 C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0 A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDA6A366AD 7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0 10B582BB67A9048FEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553 1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753 </pre>
DSA	DSA	The private key value (not displayable), the DSA prime p, subprime q, and base g. For example: <pre> VALUE: Not displayable PRIME: 2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33 C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9 4490C4837B5656776E9FFDA073EAD869B19F7E197970DBE5665E8F87F964C57 SUBPRIME: DF9048626B9AD71EF6F33081890DF9048626B9AD BASE: F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE 1362CDD0387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1 1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40 </pre>
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: <pre> VALUE: Not displayable VALUE BITS: 160 PRIME: F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D 18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC 662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61 5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805 CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E 69AD03FBFBCC5990EDDF8C1A34D607AC3B7728D7E6ABBD566A626980E0D888 C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B 95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562 BASE: 3EACCC1C25742C25924612B407869788F3236AF037B7D7EBB03C0FB6529A376 CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF 26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73 C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0 A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDA6A366AD 7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0 10B582BB67A9048FEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553 1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753 </pre>

Table 23. Information displayed in Private Key Object Details panel for RSA, DSA, Diffie-Hellman, and Elliptic Curve 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

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

4490C4837B5656776E9FFDA073EAD869B19F7E197970DBE5665E8F87F964C57

SUBPRIME:

DF9048626B9AD71EF6F33081890DF9048626B9AD

BASE:

F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A

D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE

1362CDDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1

1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40

Figure 243. 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 24. 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: <pre> KEY TYPE: DSA LOCAL: FALSE APPLICATION: Some UNIX Application PRIME BITS: 1024 PRIME: 2A5C655610E93CF27FF5B65B7FF69DDE1A4780C6D71012304869CFDFC3285F5A ED4493E75E438DD4A107CAE127AB8FC6B842A20AB4877C34166CA9D1F510EB33 C8193EA4A391526169262C9F4369274C682339DFB17B599B587F7B99B1AB37C9 4490C4837B5656776E9FFDA073EAD869B19F7E197970DBE5665E8F87F964C57 SUBPRIME: DF9048626B9AD71EF6F33081890DF9048626B9AD BASE: F21C09419230CAD25CB4C865BAF7A3FE59AAEC7D97A12D8C787C29D699F6650A D7DF6D09412C3727F4DB1F269B8C62433CCBBD52E651E5444D0A00834F6B4CCE 1362CDD387DC31501C9E4E5DBE9F42CFB8E0DB77CA121C4E612843DA035D4E1 1D4CD1CF81076A7BED411ECE6B9851936D08A5F651DC7FF3414EEB73109DFE40 </pre>
Diffie-Hellman	DH	The Diffie-Hellman prime p and base g. For example: <pre> KEY TYPE: DH LOCAL: FALSE APPLICATION: Some UNIX Application PRIME BITS: 2048 PRIME: F35F5EF1E1AC5D5289A7EB6340E41FDA18695CBBB2EB5E27BC3FA1C0FA0D215D 18F017AEA80631223A2F268304894246BE8F629BEF7DB621B1E1C5F90D00F1AC 662119D2179DC02F20966591E39079D7A621F522F29451F4663E664D830A2F61 5E51A722EE6124F102A8334B113426A86028F6DC1F0D4F05EBE4AE9F57BA6805 CE54B8C4C1866870110D3550689E435A6EDDA1FFA74D46C77C8850F7716EAF6E 69AD03FBFBCC5990EDDF8C1A34D607AC3B7728D7E6ABBD566A626980E0D888 C83661867992AF0EE415CA3B392C40D5138A18E983784676736A67D82F69D12B 95778A0CF92F752338CB811E1C68FBC04E8D9471B487C14942945AD6B345B562 BASE: 3EACCC1C25742C25924612B407869788F3236AF037B7D7EBB803C0FB6529A376 CF8161AACAA0B9C3D285D772C71B78264B56DE152B8B70975CE8B57D3EB048FF 26629B0A1756A4004418B6AED201AC6831CB0F555B4C1CA4721F96272C741F73 C439C3312C180BA67F5EAF823673904C78A6440A29A900B7F1C301C9FE9E7EB0 A7B286943B62AF22995CA15A1AC4FE3AB28C3C53629C581A97773CDA6A366AD 7EA29F4128D7EF45FC8D8C7A35FE51B87A3F14CCF0E5B3A7B7F80AB5A72EDAB0 10B582BB67A9048FFEE3631D50661E8FDC22E6754CAE46E06AC70F16667A7553 1C83C61047605D205C14E0032BC0C2E611B54AE1EF2DEFA67B4AEC8181910753 </pre>

Chapter 16. 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
- Initialize a CKDS and load DES and PKA master keys using a pass phrase

Restriction: You cannot use this utility to initialize a CKDS (and load DES and PKA master keys using a pass phrase) on the z990, z890, z9 EC, z9 BC, z10 EC and z10 BC.

On the supported hardware, the utility only loads DES and PKA master keys on the CCF. If you have a PCICC as part of the configuration, the SYM-MK is not loaded.

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 344 provides sample code.

For information about using the utility program to reencipher a disk copy of a CKDS and change the master key, see “Reenciphering a disk copy of a CKDS and changing the master key.” For information about using the program to refresh the in-storage CKDS, see “Refreshing the in-storage CKDS using a utility program” on page 341.

Reenciphering a disk copy of a CKDS and changing the master key

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 KGUP updates” on page 192. 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.

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.

Notes:

- a. 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.
 - b. 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.
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” on page 342.

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” on page 342.

Loading DES and PKA master keys using a pass phrase

This topic describes how to use the CSFEUTIL program to load DES and PKA master keys using a pass phrase. This will allow an automated setup of ICSF for an automated electronic delivery process.

Restriction: This is not supported on a z990, z890, z9 EC, z9 BC, z10 EC and z10 BC.

The CKDS must be created and empty. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for this information.

Note: This only initializes the CCF. It will not initialize the PCICC.

The default pass phrase supplied is Change this Pass Phrase.

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 name of the CKDS 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 load a pass phrase, pass these parameters in this order:
 - The name of the CKDS
 - An optional 16–64 character pass phrase
 - The name for the task: PPINIT
5. To load the pass phrase using JCL (with the default pass phrase), use JCL like this example:

```
//STEP EXEC PGM=CSFEUTIL,PARM='CSF.CSFCKDS,PPINIT'
```
6. To load the pass phrase using JCL (and using your own pass phrase), use JCL like this example:

```
//STEP EXEC PGM=CSFEUTIL,PARM='CSF.CSFCKDS,different pass phrase,PPINIT'
```

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.
72 or 104	CKDS processing has failed. A return code 72 indicates the error was detected in 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 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 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
--------------------	----------------

36000	Unable to change master key. Check hardware status.
--------------	---

36008	Crypto master key register(s) in improper state.
--------------	--

36020	Input CKDS is empty or not initialized (authentication pattern in the control record is invalid).
--------------	---

36036	The new master key register for Coprocessor 1 (C1) is not full, but C0 is ready and the current master key is valid.
--------------	--

36040	The new master key register for C0 is not full, but C1 is ready and the current master key is valid.
--------------	--

36044	The master key authentication pattern for the CKDS does not match the authentication pattern of the coprocessors, which are not equal.
--------------	--

36048	The master key authentication pattern for the CKDS does not match the authentication pattern of either of the coprocessors, which are not equal.
--------------	--

36052	A valid new master key is present in C0, but its authentication pattern does not match that of C1 or the CKDS, which are equal.
--------------	---

36056	A valid new master key is present in C1, but its authentication pattern does not match that of C0 or the CKDS, which are equal.
--------------	---

36060	The new master key register(s) is/are not full.
--------------	---

36064	Both new master key registers are full but not equal.
--------------	---

36068	The input CKDS is not enciphered under the current master key.
--------------	--

36076	The new master key register for C0 is not full, but the CPUs are online.
--------------	--

36080	The new master key register for C1 is not full, but the CPUs are online.
--------------	--

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.
--------------	---

34108	PKA callable services are enabled, and the PKDS is the active PKDS as specified in the options data set.
--------------	--

36120	The CKDS is unusable. The CKDS does not support record level authentication.
--------------	--

36124	The CKDS is unusable. The CKDS only supports encrypted AES keys and encrypted DES support is required.
--------------	--

36128	The CKDS is unusable. The CKDS does not support encrypted DES keys which is required.
--------------	---

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.

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
//* 5694-A01
//* (C) Copyright IBM Corp. 2004
//*
//*
//* 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.
//*
```

```

/** CSFEUTL: 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.
/**
/** CSFEUTL 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 CSFEUTL assembler code.
/** This CSFEUTL 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 CSFEUTIL 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 CSFEUTL 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 CSFEUTL EXEC JCL, for the PARM='....' specify the
/** requested function for CSFEUTIL.
/**
/** 8.Users may want to separate the CSFEUTL EXEC JCL into a
/** separate JOB.
/**
/** NOTES:
/** 1.This job should be rerun with every new release of ICSF.
/**
/** *****
/** JCL to assemble CSFEUTL
/** *****
/** ASSEMBLER
/**C EXEC PGM=ASMA90,REGION=4M
/**SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
/** DD DSN=SYS1.MODGEN,DISP=SHR
/**SYST1 DD DSN=SYS1.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  *
*****
* CSFEUTL assembler code *
*****

      TITLE 'CSFEUTL - 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 : CSFEUTL
*
* 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
ROSAVE  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.SCSFMODE,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 17. 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:

- Initialize a PKDS
- 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 "Reenciphering a PKDS" on page 350. 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 350.

Initializing a PKDS

You can initialize 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 name of the PKDS 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 initialize a PKDS, pass these parameters in this order:
 - a. The name of the PKDS to initialize.
 - b. The name for the task: INITPKDS.
4. To initialize the PKDS using JCL, use JCL like this example:

```
//STEP EXEC PGM=CSFPUTIL,PARM='NEW.PKDS,INITPKDS'
```

The first parameter passed, NEW.PKDS, is the name of the PKDS that you want ICSF to initialize.

Note: "CSFWPUTL" on page 352 provides sample code.

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” on page 351.

Reenciphering a 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: RECIPIHER.
4. To reencipher the PKDS using JCL, use JCL like this example:

```
//STEP EXEC PGM=CSFPUTIL,PARM='OLD.PKDS,NEW.PKDS,RECIPIHER'
```

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” on page 351.

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 342.

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 12 has this reason code:

Reason Code	Meaning
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
//*      5694-A01
//*      (C) Copyright IBM Corp. 2004
//*
//*
//* 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,PARMLN                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 CSFWPUTIL 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 18. 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 READ authority to the CSFDUTIL resource in the CSFSERV class.

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 358.

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 25. CKDS information from CSFDUTIL

Column	Value
1 - 64	Key label
67 - 74	Key type from the KDS record
77 - 84	Creation date. yyyyymmdd
87 - 94	Creation time. hhmmssst
97 - 104	Last update date. yyyyymmdd
107 - 114	Last update time. hhmmssst

The PKDS information that is written out has the format:

Table 26. PKDS information from CSFDUTIL

Column	Value
1 - 64	Key label
67 - 74	Creation date. yyyyymmdd
77 - 84	Creation time. hhmmssst
87 - 94	Last update date. yyyyymmdd
97 - 104	Last update time. hhmmssst

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 dataset or DD. See the output for details.
- 6036** There was an error deallocating a dataset or DD. See the output for details.

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
//*        5694-A01
//*        Copyright IBM Corp. 2008
//*
//*
//* 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)
/*
```

Appendix A. CCC Bit Assignments

These are some of the hardware CCC (crypto configuration control) definitions. You can view these values from the coprocessor hardware status panel (see Figure 185 on page 268). You are not able to change these values.

Note: The CCC applies only to the Cryptographic Coprocessor Feature. You do not see CCC definitions on the panel for the PCIXCC, CEX2C, or CEX3C.

<i>BIT</i>	<i>Meaning</i>
6	indicates TKE can be supported.
37 - 38	indicates triple DES and AES are supported.

Bits 80 through 127 (the right-most bits on the hardware status panel) form a pattern indicating the key length that is allowed.

When these bits are 07F7F 0F7F7, the maximum RSA key management key length is 512 bits.

When these bits are 0FFFF 0FFFF, the maximum RSA key management key length is 1024 bits.

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(s) 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 27 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. Control vectors indicated with an "*" are supported by the CCF.

Table 27. 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
*AKEK	00 00 00 00 00 00 00 00	
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
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	
*DATA	00 00 00 00 00 00 00 00	
DATAAC	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

Table 27. 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
*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
*DATAXLAT	00 06 71 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

Notes:

1. The external control vectors for DATAC, double-length MAC generation and MAC verification keys are also referred to as data compatibility control vectors.
2. Double-length MAC and MACVER keys can now be specified by these key types on the IBM @server zSeries 990, z890, z9 EC and z9 BC.

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 ICSF utility panels that are described in Chapter 6, “Managing Master Keys - CCF and PCICC,” on page 93 or the checksum algorithm that is described in this appendix.

In the checksum algorithm, you use these operations:

- Sum Operation

The addition table in Figure 244 on page 366 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 244. Addition Table

- Shift Operation

The shift table in Figure 245 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 245. 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, 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," 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.

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 master key verification pattern algorithm

The AES 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

Algorithm for calculating an authentication pattern

When you initialize a CKDS, ICSF uses the current master key and the authentication pattern algorithm to calculate an authentication pattern for the CKDS. ICSF places the value of the authentication pattern in the header record of the CKDS.

At ICSF startup, ICSF uses the authentication pattern to verify that the master key enciphers the current CKDS specified at ICSF startup. It compares the authentication pattern that is stored in the CKDS with the authentication pattern it calculates for the master key. If the authentication patterns do not match, ICSF startup fails, and ICSF gives you a message that states that the master key is not valid.

To calculate the authentication pattern, ICSF uses this algorithm:

1. Encrypt the left half of the master key under the key 6767 6767 6767 6767, using the DES algorithm.
2. Exclusive OR the result of step 1 with the original left half of the key.
3. Use the result of step 2 as the DES key in the DES algorithm to encrypt the right half of the master key.
4. Exclusive OR the result of step 3 with the original right half of the master key.

The resulting 64-bit value is the authentication pattern.

Pass Phrase Initialization master key calculations

The values for the DES and PKA 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 PKA signature master key and the last eight bytes of the calculation become the last eight bytes of the PKA key management master key.
2. ICSF generates the DES master key value by appending 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.
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 PKA signature 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 becomes the first 16 bytes of the PKA key management master key.
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.

Note: If the SMK=KMMK option is selected or defaulted, the KMMK is not used.

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. For more detailed information on configuring and running in LPAR mode, refer to the *zSeries PR/SM Planning Guide* and the *S/390 Hardware Management Console Guide*.

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 you are using.

Allocating Resources on z/990 or z890

For z9 EC, z9 BC and IBM System z10 Enterprise Class only CEX2C and CEX2A are supported.

To dynamically enable use of a new PCIXCC/CEX2C or PCICA/CEX2A coprocessor 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(s) 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 28 on page 372 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(s) 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 28. Planning LPARs domain and cryptographic coprocessor

coprocessor ID	AP0	AP1	AP2	AP3	AP4	AP5	AP6	...
type	PCICA or PCIXCC or CEX2C	PCICA or PCIXCC or CEX2C	PCICA or PCIXCC or CEX2C	PCICA or PCIXCC or CEX2C	PCICA or PCIXCC or CEX2C	PCICA or PCIXCC or CEX2C	PCICA or PCIXCC or CEX2C	
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	
.../...								

Up to 30 partitions can be defined and active, and each coprocessor has 16 domains. Within a row, the domain index number(s) specified are identical since 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.

Allocating Resources on CCF Systems

You use the Hardware Master Console tasks to enable various cryptographic functions for an LP. To assign a control domain index and usage domain index and initially enable cryptographic functions for an LP, use the Crypto page of the Customize Activation Profiles task. On the Crypto page you can enable these functions to the LP:

- Public key algorithm (PKA) function
- Cryptographic functions
 - Special secure mode
 - Public key secure cable (PKSC) and Integrated Cryptographic Service Facility (ICSF)
 - Modify authority (only enabled in one LPAR partition at a time)
 - Query signature controls
 - Query transport controls

These functions are hierarchically applied. For instance, if you do not enable cryptographic functions for the LP, you cannot enable any of the functions below it on the list. To enable basic ICSF functions, you must select these parameters on the crypto page:

- Usage domain index
The number you select for usage domain index must match the domain number that is entered in the installation options data set for this LP.
- Enable cryptographic functions
- Enable public key secure cable (PKSC) and Integrated Cryptographic Service Facility (ICSF)

Once an LP is activated, you can then use the Change LPAR Crypto task to change the cryptographic functions that are enabled for that LP. This task has a page for each LP.

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 DES, SYM-MK, ASYM-MK and PKA master keys.

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 PCIX Workstation User's Guide*.

If you are using the TSO panels, follow the procedure in “Steps for changing master keys” on page 119 or “Steps for changing master keys” on page 163 for your DES, SYM-MK, ASYM-MK and PKA master keys. Your key type should equal DES or SYM-MK and the key value should be all zeros.

```

CSFDKE10 ----- ICSF - Clear Master Key Entry -----
COMMAND ==>

                CCF DES/PCICC SYM-MK new master key register      : EMPTY
                CCF Signature/PCICC ASYM-MK master key register   : FULL
                CCF Key management master key register            : FULL

Specify information below
Key Type ==> DES          (DES, SMK, KMMK, ALL-PKA)

Part      ==> FIRST      (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
           ==> 0000000000000000
           ==> 0000000000000000 (SMK, KMMK and ALL-PKA only)

```

Figure 246. The Clear Master Key Entry Panel - CCF and PCICC

```

CSFDKE50----- ICSF - Clear Master Key Entry -----
COMMAND ==>

                Symmetric-keys new master key register           : EMPTY
                Asymmetric-keys new master key register          : FULL

Specify information below
Key Type ==> SYM-MK          (SYM-MK, ASYM-MK)

Part      ==> FIRST          (RESET, FIRST, MIDDLE, FINAL)

Checksum  ==> 00

Key Value ==> 0000000000000000
           ==> 0000000000000000
           ==> 0000000000000000 (ASYM-MK only)

Press ENTER to process.
Press END   to exit to the previous menu.

```

Figure 247. The Clear Master Key Entry Panel - PCIXCC, CEX2C, and CEX3C

Appendix E. 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 29. Callable services and parameters affected by key store policy

ICSF callable service	31-bit name	Parameter checked
ANSI X9.17 key export	CSNAKEX	source_data_key_1_identifier source_data_key_2_identifier source_key_encrypting_key_identifier transport_key_identifier
ANSI X9.17 key import	CSNAKIM	transport_key_identifier
ANSI X9.17 key translate	CSNAKTR	inbound_transport_key_identifier outbound_transport_key_identifier
ANSI X9.17 transport key	CSNATKN	source_transport_key_identifier
Cipher text translate	CSNBCTT	key_identifier_in key_identifier_out
Clear PIN encrypt	CSNBCPE	PIN_encrypting_key_identifier
Clear PIN generate alternate	CSNBCPA	PIN_encryption_key_identifier PIN_generation_key_identifier
Clear PIN generate	CSNBPGN	PIN_generation_key_identifier
Control vector translate	CSNBCVT	KEK_key_identifier source_key_token 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	source_key_token importer_key_identifier
Decipher	CSNBDEC	key_identifier
Digital signature generate	CSNDDSG	PKA_private_key_identifier

Table 29. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
Digital signature verify	CSNDDSV	PKA_public_key_identifier
Diversified key generate	CSNBDBG	generating_key_identifier generated_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
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 test extended	CSNBYTX	key_identifier kek_key_identifier
Key translate	CSNBKTR	input_key_token input_KEK_key_identifier
MAC generate	CSNBMGN	key_identifier
MAC verify	CSNBMGN	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	exporter_key_identifier

Table 29. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
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
Remote key export	CSNDRKX	trusted_block_identifier transport_key_identifier importer_key_identifier source_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
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 decipher	CSNBSYD	key_identifier
Symmetric algorithm encipher	CSNBSYE	key_identifier
Symmetric key export	CSNDSYX	DATA_key_identifier RSA_public_key_identifier

Table 29. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
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
Transaction validation	CSNBTRV	transaction_key_identifier
Transform CDMF key	CSNBTCK	source_key_identifier kek_key_identifier
Trusted block create	CSNDTBC	input_block_identifier transport_key_identifier
User derived key	CSFUDK	derivation_key_identifier source_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 30. 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 record create	CSNDKRC/CSNFKRC	token
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 31. 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 32. Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions (token)

	No KSP	KSP			
		No CSNDSYX Access Controls for algorithm	CSNDSYX Access Controls for algorithm	No Granular Keylabel Access Controls	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 F. 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 G. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

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

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

Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This ICSF Administrator's Guide is intended to help the ICSF administrator manage the cryptographic keys.

This book primarily documents information that is NOT intended to be used as a Programming Interface of OS/390 ICSF.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of OS/390 ICSF. This information is identified where it occurs, either by an introductory statement to a topic or by the following marking:

Programming Interface information

End of Programming Interface information

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	Personal System/2
ES/3090	Processor Resource/Systems Manager
IBM	PR/SM
IBMLink	RACF
MVS/DFP	S/390
MVS/ESA	S/390 Parallel Enterprise Server
Multiprise	SecureWay
OS/390	Resource Link
Personal Security	3090
zSeries	z/OS

The e-business logo is a trademark of IBM.

The following term is a trademark of another company:

American Express	American Express Company
MasterCard	MasterCard International Incorporated
Netscape	Netscape Communications Corporation
VISA	VISA International Service Association

Other company, product, and service names may be trademarks or service marks of others.

Index

A

- access control, using RACF to control use of
 - cryptographic keys and services 43
- accessibility 383
- Activate PKDS panel 115, 160
- ADD control statement
 - creating using panels 231
 - example
 - add a group of CLRDES keys 216
 - add a range of CLRDES keys 216
 - adding an entry to the CKDS 211, 216
 - creating a range of NULL keys 213
 - creating keys for key exchange 213
 - with ALGORITHM keyword 217
 - with CLEAR keyword 212
 - with CLRAES keys 217
 - with CLRAES keyword 217
 - with CLRDES keyword 216
 - with range of CLRAES keys 217
 - with TRANSKEY keyword 212
 - function 203
 - syntax 197
- administrative control function
 - displaying 257
- Administrative Control Functions panel 125, 168, 184, 193
- AES
 - key exchange using RSA key scheme 21
- ALGORITHM control statement keyword 199
- Allocation panel 230
- AMS IMPORT/EXPORT commands 225
- AMS REPRO command 225
- ANSI key-encrypting key 14
- ANSI system keys
 - use of 29
- ANSI X9.17 EDC generate callable service, controlling use of 46
- ANSI X9.17 key export callable service, controlling use of 46
- ANSI X9.17 key import callable service, controlling use of 46
- ANSI X9.17 key translate callable service, controlling use of 46
- ANSI X9.17 key transport key partial notarize 46
- ASYM-MK master key
 - initializing 152
- asymmetric master key
 - register 280
- asymmetric-keys master key
 - register 272
- AUDIT operand
 - for profiles in the CSFKEYS general resource class 45
 - for profiles in the CSFSERV general resource class 49
- authentication pattern
 - algorithm 367

- authentication pattern (*continued*)
 - description 220, 367
- Authorized UDX Coprocessor Selection panel 306
- Authorized UDX panel 308
- Authorized UDXs panel 307
- automated teller machines
 - atm
 - remote key loading 20

B

- batch LSR 224

C

- callable service, installation-defined 301
- Change Master Key panel 123, 167
- change master key panel service, controlling use of 46
- changing master keys 119, 163
- changing the master key
 - using panels 121, 165
- changing the master key using a utility program 339
- CHECKAUTH installation option 282
- checksum
 - algorithm 365
 - description 98, 139
 - general 94, 136
 - generating for master key entry 94, 135
 - generating 97, 139
- Checksum and Verification Pattern panel
 - initial 99, 140
 - requesting calculations 100, 141
 - with calculation results 100
- CIPHER macro, controlling use of 47
- ciphertext translate callable service, controlling use of 46
- CKDS
 - entering keys into 30
 - managing in a SYSPLEX environment 181
 - sharing 181
- CKDS (cryptographic key data set)
 - description 33
 - disallowing dynamic update 192
 - initializing 110, 152
 - installation option 282
 - panel option 112, 154, 157, 178, 179
 - record format 218
 - reenciphering 122, 166
 - using a utility program 339
 - refreshing
 - using a utility program 341
 - using panels 226, 250
 - specifying using panels 244
- CKDS conversion utility
 - CSFCONV 46
- CKDS/PKDS
 - migrating to a z990, z890, z9 EC or z9 BC 185

- CKTAUTH installation option 284
- CLEAR control statement keyword 201
- clear key 9
- clear key import callable service, controlling use of 46
- Clear Master Key Entry panel 374
- clear master key entry panel service, controlling use of 46
- clear PIN encrypt, controlling use of 46
- clear PIN generate alternate, controlling use of 46
- COMPAT installation option 283
- complementary key 194
- Confirm Restart Request panel 109, 126, 151
- control statement 196
 - creating using panels 227
 - editing 242
 - input data set
 - description 221
 - specifying using panels 228, 245
 - output data set
 - description 223
 - specifying using panels 245
- control vector
 - description 16, 191, 363
 - value 363
- control vector translate 46
- controlling who can use cryptographic keys and services 43
- Coprocessor Management panel 102, 125, 133, 143, 169, 174, 260, 262, 264, 265, 267, 274, 287, 290
- Coprocessors for Authorized UDX panel 307
- Coprocessors for Authorized UDXs panel 307
- CP Assist for Cryptographic Functions
 - hardware 5
- Create ADD, UPDATE, or DELETE Key Statement panel 232, 233, 235, 238
- Create RENAME Control Statement panel 239, 240
- Create SET Control Statement panel 241, 242
- crypto configuration control
 - displaying status 273
- Crypto Express2 Accelerator
 - hardware 5
- Crypto Express2 Feature
 - hardware 4
- Crypto Express3 Feature
 - hardware 4
- Cryptographic Coprocessor Feature 15
- cryptographic domain 268, 275
- cryptographic variable encipher, controlling use of 46
- CRYPTOZ class 321
- CSFDIAG data set 221
 - DD statement for 249
- CSFDUTIL utility
 - reason codes 358
- CSFDUTIL utility program
 - description 357
 - return codes 358
- CSFEUTIL utility
 - reason codes 342
- CSFEUTIL utility program
 - description 339
 - return codes 342, 351

- CSFEUTIL utility program (*continued*)
 - using 341
- CSFKEYS general resource class
 - defining profiles 45
- CSFPUTIL utility
 - reason codes 351
- CSFPUTIL utility program
 - description 349
 - using 350
- CSFSERV class
 - resources for token services 322
- CSFSERV general resource class
 - defining profiles 46

D

- data key export callable service, controlling use of 47
- data key import, controlling use of 47
- data protection 22
- data-encrypting key 10
- data-translation key 11
- Deactivate Last Coprocessor panel 266
- decipher callable service, controlling use of 46
- decode callable service, controlling use of 46
- Decode panel 311
- decoding 310
- Default Role Status Display panel 288, 289, 291, 292, 293
- DELETE control statement
 - creating using panels 231
 - example 215
 - with CLRAES key labels 217
 - with CLRAES keyword 217
 - with CLRDES key labels 216
 - with CLRDES keyword 216
 - function 215
 - syntax 209
- DES
 - key exchange using RSA key scheme 21
 - remote key loading 20
- DES control statement keyword 202
- DES master key
 - initializing 110
- diagnostics data set
 - description 221
 - specifying using panels 245
- digital signature generate callable service, controlling use of 47
- digital signature verify callable service, controlling use of 47
- disability 383
- disabling PKA callable services 93, 135
- disallowing dynamic CKDS update 192
- displaying
 - administrative control function 257
 - hardware status 266, 273
 - installation exits 293, 294
 - installation option 280
 - installation-defined callable service 301
 - installation-defined callable services 301
- distributing cryptographic keys 36

- Diversified Key Generate 46
- domain
 - reassigning 373
- DOMAIN installation option 283
- domain, cryptographic 268, 275
- DSS 15
 - key pair generation 15
- dynamic CKDS
 - update services, entering keys 31
 - update, disallowing 192
- DYNAMIC installation option 258

E

- Edit Control Statement panel 243
- editing control statement 242
- EMK macro, controlling use of 47
- encipher callable service, controlling use of 47
- encode callable service, controlling use of 47
- Encode panel 310
- encoding 309
- encrypted key 9
- encrypted PIN Generate, controlling use of 47
- entering
 - final key part manually 105, 147
 - intermediate key parts 104, 145
 - keys into the CKDS 30
 - using the dynamic CKDS update services 31
 - using the key generator utility program 31
 - keys into the PKDS 32
 - keys into the TKDS 32
- environment control mask
 - displaying status 273
- even parity
 - random numbers 97, 138
- exit
 - identifier on ICSF/MVS 298
- exits
 - displaying 293
- exportable form 19
- exporter key-encrypting key 13
- extended system keys 30

F

- factorization problem 4
- FIPSMODE installation option 286

G

- general resource class
 - CSFKEYS 45
 - CSFSERV 46
- general resource profile
 - CSF1DMK 49
 - CSF1DVK 49
 - CSF1GAV 49
 - CSF1GKP 49
 - CSF1GSK 49
 - CSF1HMG 49
 - CSF1HMV 49

general resource profile *(continued)*

- CSF1PKS 49
- CSF1PKV 49
- CSF1SAV 49
- CSF1SKD 49
- CSF1SKE 49
- CSF1TRC 49
- CSF1TRD 49
- CSF1TRL 49
- CSF1UWK 49
- CSF1WPK 49
- CSFAEGN 46
- CSFAKEX 46
- CSFAKIM 46
- CSFAKTR 46
- CSFATKN 46
- CSFBDBG 46
- CSFCKI 46
- CSFCKM 46
- CSFCMK 46
- CSFCPA 46
- CSFCPE 46
- CSFCSG 46
- CSFCSV 46
- CSFCTT 46
- CSFCTT1 46
- CSFCVE 46
- CSFCVT 46
- CSFDCO 46
- CSFDEC 46
- CSFDEC1 46
- CSFDKCS 46
- CSFDKF 46
- CSFDKM 47
- CSFDKX 47
- CSFDSG 47
- CSFDSV 47
- CSFECO 47
- CSFEDC 47
- CSFEMK 47
- CSFENC 47
- CSFENC1 47
- CSFEPG 47
- CSFGKC 47
- CSFIQF 47
- CSFKEX 47
- CSFKGN 47
- CSFKIM 47
- CSFKRC 47
- CSFKRD 47
- CSFKRR 47
- CSFKRW 47
- CSFKTR 47
- CSFKYT 47
- CSFKYTX 47
- CSFMDG 47
- CSFMDG1 47
- CSFMGN 47
- CSFMGN1 47
- CSFMVR 47
- CSFMVR1 47

general resource profile *(continued)*

CSFOWH 47
CSFOWH1 47
CSFPCAD 47
CSFPCI 47
CSFPCU 47
CSFPEX 48
CSFPEXX 48
CSFPGN 48
CSFPKD 48
CSFPKDR 48
CSFPKE 48
CSFPKG 48
CSFPKI 48
CSFPKRC 48
CSFPKRD 48
CSFPKRR 48
CSFPKRW 48
CSFPKSC 48
CSFPKT 48
CSFPKTC 48
CSFPKX 48
CSFPMCI 48
CSFPTR 48
CSFPVR 48
CSFREFR 48
CSFRENC 48
CSFRKD 48
CSFRKL 48
CSFRKX 48
CSFRNG 48
CSFRNGL 48
CSFRSWS 48
CSFRTC 48
CSFSAD 48
CSFSAD1 48
CSFSAE 48
CSFSAE1 48
CSFSBC 48
CSFSBD 48
CSFSKI 48
CSFSKM 48
CSFSKY 48
CSFSMK 49
CSFSPN 49
CSFSSWS 49
CSFSYG 49
CSFSYI 49
CSFTBC 49
CSFTCK 49
CSFTRV 49
CSFUDK 49

generating checksums, verification patterns, and hash patterns 97, 139

generating cryptographic keys 25

generating master key data 94, 135

generating PKA keys 26

GENKEY macro, controlling use of 47

Group Label Panel 237

H

hardware status

displaying 266, 273

Hardware Status Display panel 90, 268, 275

hash pattern

description 95, 136

for old master key 272, 280

generating 97, 139

I

ICSF (Integrated Cryptographic Service Facility)

description 1

importable form 19

importer key-encrypting key 14

initial transport key pair

description 195

establishing 251, 253, 255

initialization

by pass phrase 73

PCICC 82, 84

Initialize a CKDS panel 112, 116, 154, 157, 161, 178, 179

Initialize a PKDS panel 115, 159

initialize PKDS using a utility program 349

initializing the CKDS 110, 152

initializing the PKDS 152

input PIN-encrypting key 13

Installation Defined Services panel 302

installation exits

See also exits

displaying 294

Installation Exits Display panel 295, 296, 297, 298

installation option

displaying 280

Installation Option Display panel 282

installation option keyword

COMPAT 283

DOMAIN 283

Installation Options 302

Installation Options panel 281, 294

installation-defined callable services

displaying 301

INSTDATA control statement keyword 210

Integrated Cryptographic Service Facility

See ICSF (Integrated Cryptographic Service Facility)

K

KCS #11 private key sign 49

Key Administration panel 227, 244, 246

Key Administration Panel 250

KEY control statement keyword 202

key export callable service, controlling use of 47

key generate callable service 27

key generate callable service, controlling use of 47

Key generation utility program 47

key import callable service, controlling use of 47

key output data set

description 222

- key output data set *(continued)*
 - specifying using panels 245
- key part
 - description 94, 136
 - generating 95, 137
- key part import callable service, controlling use of 47
- key protection 16
- key record create callable service, controlling use of 47
- key record delete callable service, controlling use of 47
- key record read callable service, controlling use of 47
- key record write callable service, controlling use of 47
- key separation 15
- key test callable service, controlling use of 47
- key test extended callable service, controlling use of 47
- key translate, controlling use of 47
- Key Type Selection panel 99, 141, 234, 239
- key types 9
 - migrating from PCF key types 18
 - TYPE control statement keyword 199
- key-encrypting key variant
 - See transport key, variant
- KEYAUTH installation option 284
- keyboard 383
- KGUP (key generator utility program)
 - control statement
 - See control statement
 - data set 218
 - specifying using panels 243
 - description 191
 - entering keys 31
 - executing using panels 226
 - generating keys 26
 - JCL for submitting 223
 - maintaining keys 33
 - panel option 226
 - reducing control area and interval splits 225
 - return codes
 - described in explanation of message CSFG0002 224
 - running with Batch LSR 224
 - submitting JCL
 - using panels 246
- KGUP Control Statement Data Set Specification panel 228, 229
- KGUP control statement keyword
 - ALGORITHM 199
 - CLEAR 201
 - DES 202
 - KEY 202
 - LABEL 198, 209, 210
 - LENGTH 202
 - NOCV 201
 - OUTTYPE 199
 - RANGE 198, 210
 - TRANSKEY 200
 - TYPE 198, 209, 210, 214
- KGUP Control Statement Menu panel 238, 241, 242

L

- LABEL (*key-label* control statement keyword 211
- LABEL control statement keyword 198, 209, 210
- loading a pass phrase using a utility program
 - using CSFEUTIL utility program 341
- loading DES and PKA master keys
 - using CSFEUTIL utility program 341
- logical partition 371
- LPAR 371

M

- MAC (message authentication code)
 - keys 11
- MAC generate callable service, controlling use of 47
- MAC generation key 11
- MAC verification key 12
- MAC verify callable service, controlling use of 47
- master key
 - changing
 - using a utility program 339
 - concept 15
 - description 15
 - entering on the IBM @server zSeries 990 135
 - entering on the PCI Cryptographic Coprocessor 93
 - entering on the S/390 Enterprise Servers and the S/390 Multiprise 93
 - panel option 74, 82, 84
 - variant 16, 191
- master key (ASYM-MK)
 - initializing 152
- master key (DES)
 - initializing 110
- master key (SYM-MK)
 - initializing 152
- master key data
 - generating 94, 135
- Master Key Entry panel 103, 104, 105, 106, 107, 108, 109, 126, 127, 128, 133, 145, 146, 147, 148, 149, 151, 152, 170, 171, 175
- Master key management panel 112, 114, 154, 156, 158, 159, 178, 179, 185
- Master Key Management panel 118, 121, 129, 130, 163, 166, 172, 173, 231
- Master Key Values from Pass Phrase panel
 - initial 131
- master keys
 - changing 119, 163
 - clearing 132, 174
 - description 10
 - entering using the pass phrase initialization utility 73
- MDC generate callable service, controlling use of 47
- MDC-4 hash pattern
 - algorithm 368
- Member Selection List panel 230
- migrating to a z990
 - sharing a CKDS/PKDS 185
 - CCF only system 186
 - CCF with PCICCs 187

multiple encipherment 17

N

new master key register 269, 271, 276, 278, 279
NOCV
 flag 202
 processing 201, 203
NOCV control statement keyword 201, 211
NOCV-enablement key 113, 202
NOCV-enablement keys
 use of 29
non-odd parity
 random numbers 97, 138
NOSSM parameter 223
Notices 385
NOTIFY operand
 for profiles in the CSFKEYS general resource
 class 45
 for profiles in the CSFSERV general resource
 class 49

O

odd parity
 random numbers 97, 138
 required for master key 97, 138
old master key register 270, 271, 276, 278, 279
one-way hash generate (with ALET) callable service,
 controlling use of 47
one-way hash generate callable service, controlling use
 of 47
operational form 16
OPKYLOAD control statement
 example 215
 syntax 211
output PIN-encrypting key 13
OUTTYPE control statement keyword 199

P

panels

CSF@PRIM — Primary Menu 74, 83, 85, 96, 101,
111, 114, 124, 137, 142, 153, 156, 159, 168, 193,
226, 258, 259, 262, 281, 287, 290, 294, 302, 305,
309
CSFACF00 — Administrative Control
 Functions 125, 168, 258
CSFACF00 —Administrative Control Functions 184,
193
CSFCKD00 — Initialize a CKDS 112, 116
CSFCKD10 — Initialize a CKDS 154, 161, 178, 179
CSFCKD20 — Initialize a CKDS 154, 157
CSFCMK10 — Reencipher CKDS 122, 166
CSFCMK11 —Reencipher PKDS 130, 173
CSFCMK20 — Change Master Key 123, 167
CSFCMK21 — Refresh PKDS 130
CSFCMK21 —refresh PKDS 115, 160, 173
CSFCMK30 — Initialize a PKDS 115, 159
CSFCMP00 — Coprocessor Management 102,
125, 133, 260, 264, 267, 287

panels (*continued*)

CSFCMP10 — Hardware Status Display 90, 268
CSFCMP30 —Status Display 288, 289, 293
CSFCMP30 —Status Displayed for a system without
 TKE connected 291, 292
CSFCMP40 — Hardware Status Display 275
CSFCMP60 — Deactivate Last Coprocessor 266
CSFCSE10 — Create ADD, UPDATE, or DELETE
 Key Statement 232, 233, 235, 238
CSFCSE11 — Group Label Panel 237
CSFCSE12 — Key Type Selection 234, 239
CSFCSE20 — Create RENAME Control
 Statement 239, 240
CSFCSE30 — Create SET Control Statement 241,
242
CSFCSM00 — KGUP Control Statement
 Menu 231, 238, 241, 242
CSFDKE10 — Clear Master Key Entry 374
CSFDKE10 — Master Key entry 102
CSFDKE10 — Master Key Entry 103, 104, 105,
106, 107, 108, 109, 126, 127, 128, 133, 148
CSFDKE40 — Confirm Restart Request 109, 126
CSFDKE50 — Master Key Entry 145, 146, 147,
149, 151, 152, 170, 171, 175
CSFDKE50 —Clear Master Key Entry 374
CSFDKE50 —Master Key entry 143, 144
CSFDKE80 — Confirm Restart Request 151
CSFECO00 — Decode 311
CSFECO00 — Encode 310
CSFGCMP0 — Coprocessor Management 143,
169, 174, 262, 265, 274, 290
CSFMKM00 — Initialize a CKDS 112, 114, 154,
156, 158, 159, 178, 179, 185
CSFMKM00 — Master Key Management 118, 121,
129, 130, 163, 166, 172, 173
CSFMKV00 — Checksum and Verification
 Pattern 99, 100, 140, 141
CSFMKV10 — Key Type Selection 99, 141
CSFPKY00 - ICSF PKDS Keys Panel 314
CSFPKY01 - PKDS Key Request Successful 316
CSFPKY02 - PKDS Key Request Failed 318
CSFPKY03 - PKDS Public Key Export
 Successful 317
CSFPKY04 - PKDS Public Key Export Failure 318
CSFPKY05 - PKDS Public Key Import
 Successful 317
CSFPKY06 - PKDS Public Key Import Failure 319
CSFPMC00 — Pass Phrase MK/CKDS/PKDS
 Initialization 75, 88
CSFPMC00 — Pass Phrase MK/KDS
 Initialization 76, 83, 84
CSFPMC10 — Pass Phrase MK/CKDS/PKDS
 Initialization 77, 78, 86, 87
CSFPMC210 — Pass Phrase MK/CKDS/PKDS
 Initialization 79, 82
CSFPMC30 — Pass Phrase MK/CKDS/PKDS
 Initialization 80, 81
CSFPPM00 —Master Key Values from Pass
 Phrase 131
CSFRNG00 — Random Number Generator 97, 138

panels (continued)

- CSFSAE10 — KGUP Control Statement Data Set Specification 228, 229
- CSFSAE11 — Allocation 230
- CSFSAE12 — Member Selection List 230
- CSFSAE20 — Specify KGUP Data Sets 244, 246
- CSFSAE30 — Set KGUP JCL Card 247
- CSFSAE40 — Refresh In-storage CKDS 251
- CSFSAM00 — Key Administration 227, 244, 246, 250
- CSFSOP00 — Installation Options 281, 294, 302
- CSFSOP10 — Installation Option Display 282
- CSFSOP30 — Installation Exits Display 295, 296, 297, 298
- CSFSOP40 — Installation Defined Services 302
- CSFTBR00 - ICSF Token Management - Main Menu Panel 324
- CSFTBR01 - ICSF Token Management - PKCS11 Token Create Successful Panel 324
- CSFTBR02 - ICSF Token Management - PKCS11 Token Delete Confirmation Panel 324
- CSFTBR03 - ICSF Token Management - PKCS11 Token Delete Successful Panel 325
- CSFTBR04 - ICSF Token Management - PKCS11 Object Delete Successful Panel 325
- CSFTBR10 - ICSF Token Management - List Token Panel 325
- CSFTBR20 - ICSF Token Management - Token Details Panel 326
- CSFTBR30 - ICSF Token Management - Certificate Object Details Panel 328
- CSFTBR31 - ICSF Token Management - Public Key Object Details Panel 330
- CSFTBR32 - ICSF Token Management - Private Key Object Details Panel 333, 334
- CSFTBR33 - ICSF Token Management - Secret Key Object Details Panel 329
- CSFTBR34 - ICSF Token Management - Data Object Details Panel 327
- CSFTBR41 - ICSF Token Management - Domain Parameters Object Details Panel 336
- CSFUDX00 306
- CSFUDX10 306
- CSFUDX20 307
- CSFUDX30 307
- CSFUDX40 307
- CSFUDX50 308
- CSFUTL00 - PKDS 314, 323
- CSFUTL00 — Utilities 96, 98, 131, 138, 139, 310, 311
- ISREDDE — Edit Control Statement 243

parity

- random numbers 97, 138

pass phrase initialization 73

- calculations 368
- in a SYSPLEX 182

pass phrase initialization utility

- initializing 91
- initializing multiple systems 91
- SAF protection 73

pass phrase master key/CKDS initialization panel

- service, controlling use of 48

Pass Phrase MK/CKDS/PKDS Initialization panel 75, 77, 78, 79, 80, 81, 82, 86, 87, 88

Pass Phrase MK/KDS Initialization panel 76, 83, 84

PCI Cryptographic Accelerator

- hardware 5

PCI Cryptographic Coprocessor

- hardware 6
- status 260, 269

PCI interface, controlling use of 47

PCI X Cryptographic Coprocessor

- hardware 5
- status 263, 276

PCICC

- adding after CCF initialization 132

PCICC initialization 82, 84

PCIXCC/CEX2C

- adding after initialization 174

PIN (personal identification number)

- keys 12

PIN Change/Unblock 47

PIN generate callable service, controlling use of 48

PIN generation key 12

PIN translate callable service, controlling use of 48

PIN verification key 12

PIN verify callable service, controlling use of 48

PKA callable services

- disabling when entering ASYM-MK 135
- disabling when entering PKA master keys 93

PKA key decrypt callable service, controlling use of 48

PKA key encrypt callable service, controlling use of 48

PKA key generate, controlling use of 48

PKA key import callable service, controlling use of 48

PKA key token change 48

PKA key translate callable service, controlling use of 48

PKAcall installation option 259

PKCS #11 derive key 49

PKCS #11 derive multiple keys 49

PKCS #11 generate HMAC 49

PKCS #11 generate key pair 49

PKCS #11 generate secret key 49

PKCS #11 get attribute value 49

PKCS #11 public key verify 49

PKCS #11 secret key decrypt 49

PKCS #11 secret key encrypt 49

PKCS #11 set attribute value 49

PKCS #11 token record create 49

PKCS #11 token record delete 49

PKCS #11 token record list 49

PKCS #11 unwrap key 49

PKCS #11 verify HMAC 49

PKCS #11 wrap key 49

PKCS11 TOKEN panel 323

PKDS

- activating 129, 172
- entering keys into 32
- initializing
 - using a utility program 349
- installation option 282

- PKDS (*continued*)
 - managing 36
 - managing in a SYSPLEX environment 183
 - reenciphering 129, 172
 - using a utility program 350
 - refreshing
 - using a utility program 350
 - using Master Key Management panel 184
- PKDS (cryptographic key data set)
 - initializing 152
- PKDS (PKA key data set)
 - panel option 115, 159
- PKDS activate 48
- PKDS panel 314, 316, 317, 318, 319
- PKDS reencipher panel service 48
- PKDS Write Create and Delete installation option 259
- PKSC interface, controlling use of 48
- PPINIT recovery 88
- PPINITmigration 87
- PR/SM consideration
 - entering
 - keys into the KSU 373
 - the master key 373
- primary menu panel 74, 75, 83, 85
- Primary Menu panel 74, 83, 85, 96, 101, 111, 114, 124, 137, 142, 153, 156, 159, 168, 193, 226, 259, 262, 281, 287, 290, 294, 302, 305, 309
- prohibit export extended callable service, controlling use of 48
- prohibit export, controlling use of 48
- protecting
 - data 22
 - keys sent between systems 20
 - keys stored with a file 19

R

- RACF
 - sample commands
 - ADDGROUP 44
 - ALTUSER 44
 - CONNECT 44
 - PERMIT 46, 50
 - RDEFINE 45, 46
 - REMOVE 44
 - SETOPTS 46, 50
 - using to control use of cryptographic keys and services 43
- random number generate callable service, controlling use of 48
- random number generate long callable service, controlling use of 48
- Random Number Generator panel 97, 138
- random numbers
 - parity 97, 138
- RANGE control statement keyword 198, 210
- reason codes
 - CSFDUTIL utility 358
 - CSFEUTIL utility 342
 - CSFPUTIL utility 351
- REASONCODES installation option 285

- Reencipher CKDS panel 122, 166
- reencipher CKDS panel service, controlling use of 48
- Reencipher PKDS panel 130, 173
- reenciphering a PKDS using a utility program
 - using CSFPUTIL utility program 350
- reenciphering CKDS using a utility program 339
- reenciphering in-storage CKDS using a utility program
 - using CSFEUTIL utility program 341
- reenciphering PKDS using a utility program 350
- refresh CKDS panel service, controlling use of 48
- Refresh In-storage CKDS panel 251
- refresh PKDS panel 130, 173
- refreshing the CKDS
 - using panels 115, 160, 226, 250
- refreshing the in-storage CKDS
 - using CSFEUTIL utility program 341
- refreshing the in-storage copy of the PKDS
 - using CSFPUTIL utility program 350
- refreshing the PKDS
 - using Master Key Management panel 184
- remote key export callable service, controlling use of 48
- RENAME control statement
 - creating using panels 238
 - example 215
 - with CLRAES keyword 217
 - with CLRDES keyword 217
 - syntax 209
- restarting the key entry process 108, 150
- retained key 15, 26
- retained key delete callable service, controlling use of 48
- retained key list callable service, controlling use of 48
- RETKEY macro, controlling use of 48
- return codes
 - CSFDUTIL utility 358
 - CSFEUTIL utility 342, 351
 - KGUP
 - described in explanation of message CSFG0002 224
- reusing a domain 373
- RSA 15
- RSA encrypted data keys
 - exchanging 19
 - key exchange 19
- RSA protected AES key exchange 21
- RSA protected DES key exchange 21

S

- secure key import callable service, controlling use of 48
- secure messaging for keys 48
- secure messaging for PINs 49
- security
 - using RACF to control use of cryptographic keys and services 43
- SERNBR control statement keyword 211
- service
 - installation-defined 301

- SET control statement
 - creating using panels 240
 - example 215
 - syntax 210
- Set KGUP JCL Card panel 247
- set master key panel service, controlling use of 49
- setting the ASYM-MK master key 152
- setting the DES master key 110
- setting the DES-MK master key 152
- setting up the PKDS 36
- sharing a CKDS/PKDS
 - migrating to a z990 185
 - CCF only system 186
 - CCF with PCICCs 187
- shortcut keys 383
- SINGLE control statement keyword 202
- SO R/W
 - description 322
- special secure mode
 - CLEAR control statement keyword 201
 - displaying status 272
 - KGUP considerations 31
 - SSM or NOSSM parameter for KGUP 223
 - submitting KGUP job stream using panel 248
- Specify KGUP Data Sets panel 244, 246
- SSM
 - installation option 284
 - parameter 223
- status
 - Cryptographic Coprocessor 260, 269, 276
 - installation exits 294
 - installation-defined callable services 301
 - panel option 257, 280
 - PCI Cryptographic Coprocessor 260, 269
 - PCI X Cryptographic Coprocessor 263, 276
 - viewing 257
- Strong SO
 - description 322
- SYM-MK master key
 - initializing 152
- symmetric key export callable service, controlling use of 49
- symmetric key generate callable service, controlling use of 49
- symmetric key import callable service, controlling use of 49
- symmetric master key
 - register 277, 279
- symmetric-keys master key
 - register 271
- SYSPLEX
 - managing the CKDS 181
 - managing the PKDS 183
 - managing the TKDS 189
 - setting DES master keys 181
 - using pass phrase initialization 182
- SYSPLEXCKDS installation option 285
- SYSPLEXTKDS installation option 286
- system keys
 - entering into the CKDS 29

T

- TKDS
 - entering keys into 32
 - installation option 282
 - managing in a SYSPLEX environment 189
- TKDS key protection 26
- TKDS panel 324, 325, 326, 327, 328, 329, 330, 333, 334, 336
- token
 - access levels 322
- TRACEENTRY installation option 285
- transaction validation 49
- transform CDMF key callable service, controlling use of 49
- TRANSKEY control statement keyword 200
- transport key
 - description 13, 15
 - initial pair 195, 251, 253, 255
 - use 194
 - variant 17
- trusted block create callable service, controlling use of 49
- TYPE control statement keyword 198, 209, 210
- type of key 9

U

- UDX Options Menu panel 306
- UPDATE control statement
 - creating using panels 231
 - example 215
 - with ALGORITHM keyword 218
 - with CLRAES keyword 217
 - with CLRDES keyword 216
 - function 203
 - syntax 197
- user control functions (TSO panel) 48, 49
- user control functions display panel 258
- User R/O
 - description 322
- User R/W
 - description 322
- USERPARM installation option 285
- using ANSI system keys 29
- using NOCV-enablement keys 29
- using RSA encryption 19
- Utilities panel 96, 98, 131, 138, 139, 310, 311
- utility panel option 95, 137, 309
- utility program 339, 349, 357
 - to change the master key 339
 - to initialize a PKDS 349
 - to reencipher a CKDS 339
 - to reencipher a PKDS 350

V

- verification pattern
 - algorithm 367
 - description 94, 95, 135, 136
 - for asymmetric master key 272, 280

- verification pattern (*continued*)
 - for final key part 107, 149
 - for new master key part 107, 149
 - generating 97, 139
- viewing system status 257

W

- WAITLIST installation option 286
- Weak SO
 - description 322
- Weak User
 - description 322

Readers' Comments — We'd Like to Hear from You

z/OS
Cryptographic Services
Integrated Cryptographic Service Facility
Administrator's Guide

Publication No. SA22-7521-14

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: mhvrfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address



Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape



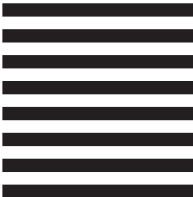
NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5647-A01

Printed in USA

SA22-7521-14

