z/OS

IBM

Cryptographic Services
Integrated Cryptographic Service Facility
CCA and PKCS #11 Algorithm Currency
APAR OA61253

Table of Contents

Chapter 1. Overview

This document describes changes to the Integrated Cryptographic Service Facility (ICSF) product in support of the following changes for CCA and PKCS #11 algorithm currency:

- Enhancement for CCA services
  - Enhancements for German Banking Industry Committee (DK)
  - New service Encrypted PIN Verify2 (CSNBPVR2 and CSNEPVR2)
  - Australian Payment Network enhancements in support of standard AS2805.5.4
  - Support for the Schnorr digital signature algorithm
  - Support for key exchange with Azure Cloud services.
  - New key usage values for AES CIPHER keys in Key Generator Utility Program
- Enhancements to PKCS #11 services
  - New service PKCS #11 Secret Key Reencrypt (CSFPSKR and CSFPSKR6).
  - Enhancements for Koblitz elliptic curves.

These changes are available through the application of the PTF for APAR OA61253 and apply to FMID HCR77D1 and HCR77D2.

This document contains alterations to information previously presented in the following books:

- z/OS Cryptographic Services ICSF Application Programmer’s Guide, SC14-7508-10
- z/OS Cryptographic Services ICSF Administrator's Guide, SC14-7506-1
- z/OS Cryptographic Services ICSF System Programmer’s Guide, SC14-7507-10
- z/OS Cryptographic Services ICSF Overview SC14-7505-10
- z/OS Cryptographic Services ICSF Writing PKCS #11 Applications SC14-7510-08

The technical changes made to the ICSF product by the application of the PTF for APAR OA61253 are indicated in this document by red text.

Chapter 2. Update of z/OS Cryptographic Services ICSF Overview, SC14-7505-10, information

This topic contains updates to the document z/OS Cryptographic Services ICSF Overview, SC14-7505-09, for the updates provided by this APAR. Refer to this source document if background information is needed.

Summary of callable service support by hardware configuration

Table 1. Services that require a CCA coprocessor

Service Name	**Function**	**A**	**B**	**C**	**D**	**E**	**F**
Encrypted PIN Verify2	Compares a supplied PIN against a reference PIN in encrypted PIN blocks.						X

Table 2. Summary of PKCS #11 callable services support

Service Name	Function
PKCS #11 Secret Key Reencrypt	Decrypts and re-encrypts data using secure secret keys

Chapter 3. Update of z/OS Cryptographic Services ICSF Administrator's Guide, SC14-7506-10, information

This topic contains updates to the document z/OS Cryptographic Services ICSF Administrator's Guide, SC14-7506-04, for the updates provided by this APAR. Refer to this source document if background information is needed.

Managing cryptographic keys

Table 3. Key Store Policy controls

The following Key Store Policy Controls	**Consists of the following XFACILIT class discrete profiles:**	**Description:**
Archived Key for Data Decryption Use control Specifies that ICSF allows an application to use the key material of a CKDS or TKDS record that has been archived for only data decrypt operations	CSF.KDS.KEY.ARCHIVE.DATA.DECRYPT	Enables the Archived Key for Data Decryption Use control. The **Key Archive Use** control need not be active. When this control is enabled, an archived data-encryption key is allowed to be used in a service that does data decryption. The key is allowed to be use with these services that do data decryption. These services do data decryption: • Decipher (CSNBDEC, CSNEDEC, CSNBDEC1, and CSNEDEC1). • Ciphertext Translate2 (CSNBCTT2, CSNECTT2, CSNBCTT3, and CSNECTT3): – Inbound key identifier . • Symmetric Algorithm Decipher (CSNBSAD, CSNESAD, CSNBSAD1, and CSNESAD1). • Symmetric Key Decipher (CSNBSYD, CSNESYD, CSNBSYD1, and CSNESYD1). • Field Level Decipher (CSNBFLD and CSNEFLD). • FPE Decipher (CSNBFPED and CSNEFPED). • FPE Translate (CSNBFPET and CSNEFPET): – Inbound key identifier. • Format Preserving Algorithms Decipher (CSNBFFXD and CSNEFFXD. • Format Preserving Algorithms Translate (CSNBFFXT and CSNEFFXT): – Inbound key identifier.

		• PKCS #11 Secret key decrypt (CSFPSKD and CSFPSKD6). • PKCS #11 Secret Key Reencrypt (CSFPSKR and CSFPSKR6) – Inbound (decryption) key object When this control is enabled, an archived data encryption key is not allowed to be used in a service that does data encryption. The service request fails with these services that do data encryption: • Encipher (CSNBENC, CSNEENC, CSNBENC1, and CSNEENC1). • Ciphertext Translate2 (CSNBCTT2, CSNECTT2, CSNBCTT3, and CSNECTT3): – Outbound key identifier. • Symmetric Algorithm Encipher (CSNBSAE, CSNESAE, CSNBSAE1, and CSNESAE1). • Symmetric Key Encipher (CSNBSYE, CSNESYE, CSNBSYE1, and CSNESYE1). • Field Level Encipher (CSNBFLE and CSNEFLE). • FPE Encipher (CSNBFPEE and CSNEFPEE). • FPE Translate (CSNBFPET and CSNEFPET): – Outbound key identifier. • Format Preserving Algorithms Encipher (CSNBFFXE and CSNEFFXE). • Format Preserving Algorithms Translate (CSNBFFXT and CSNEFFXT): – Outbound key identifier. • PKCS #11 Secret Key Encrypt (CSFPSKE and CSFPSKE6). • PKCS #11 Secret Key Reencrypt (CSFPSKR and CSFPSKR6) – Outbound (encryption) key object

Using KGUP control statements

Syntax of the ADD and UPDATE control statements

KEYUSAGE(key-usage-value[,key-usage-value]...)

This keyword defines key usage values for the key that is being generated. The usage values are used to restrict a key to a specific algorithm or usage.

The associated data for variable length tokens is described in Appendix B of the Application Programmer's Guide. The DES control vector is described in Appendix C. of the Application Programmer's Guide.

The following values have been defined. The usage values are specific to a key type. The values can only be specified for the key type that is indicated in the following tables.

Note: Any value with a non-alphanumeric character must be enclosed in quotes when specified with the KEYUSAGE keyword. For example:

KEYUSAGE('CVVKEY-A')

When a pair of keys is generated, one for the local system and the other for a remote system, both keys are generated with the same key-usage flags when the KEYUSAGE keyword is used.

Table 4. Usage values for key types

Key type	Key algorithm	Key Usage Values
CIPHER	AES	The following values are optional: C-XLATE, V1PYLD **and** One of following value is optional: ANY-MODE, FF1, FF2, FF2.1, GCM **and** One or both can be specified: DECRYPT, ENCRYPT. **Note**: The key generated when KEYUSAGE is not specified has only the DECRYPT and ENCRYPT key-usage. This is the default. **Note**: When no encryption mode keyword is specified, the encryption mode will default to CBC.

Table 5. Meaning of usage values

Key Usage Value	Key types	Meaning
ANY-MODE	CIPHER	This key can be used for any encryption mode.
FF1	CIPHER	This key can be used for Format Preserving method FF1.
FF2	CIPHER	This key can be used for Format Preserving method FF2
FF2.1	CIPHER	This key can be used for Format Preserving method FF2.1.
GCM	CIPHER	This key can be used for Galois/counter mode.

Callable services affected by key store policy

Table 6. Callable services and parameters affected by key store policy		
ICSF callable service	**31-bit name**	**Parameter checked**
Encrypted PIN Verify2	CSNBPVR2	input_PIN_encrypting_key_identifier reference_PIN_encrypting_key_identifier
Random Number Generate Long	CSNBRNGL	key_identifier
Symmetric Algorithm Encipher	CSNBSAE	key_identifier key_parms when the parameter is used to pass a key identifier to the service

Callable services that trigger reference date processing

Table 7. Callable services and parameters that trigger reference date processing		
ICSF callable service	**31-bit name**	**Parameter checked**
Encrypted PIN Verify2	CSNBPVR2	input_PIN_encrypting_key_identifier reference_PIN_encrypting_key_identifier
Random Number Generate Long	CSNBRNGL	key_identifier
Symmetric Algorithm Encipher	CSNBSAE	key_identifier key_parms when the parameter is used to pass a key identifier to the service
PKCS #11 Secret Key Reencrypt	CSFPSKR	decrypt_key_handle encrypt_key_handle

Resource names for CCA and ICSF entry points

Table 8. Resource names for CCA and ICSF entry points						
Descriptive service name	**CCA entry point name**		**ICSF entry point name**		**SAF resource name**	**Callable service exit name**
Encrypted PIN Verfiy2	CSNBPVR2	CSNBEPVR2	CSFPVR2	CSFPVR26	CSFPVR2	CSFPVR2
PKCS #11 Secret Key Reencrypt	N/A	N/A	CSFPSKR	CSFPSKR6	CSF1SKR[1]	N/A

Notes:

– [1] CSF1xxx is just another name for the CSFPxxx service.

Chapter 4. Update of z/OS Cryptographic Services ICSF System Programmer's Guide, SC14-7507-10, information

This topic contains updates to the document z/OS Cryptographic Services ICSF System Programmer's Guide, SC14-7507-09, for the updates provided by this APAR. Refer to this source document if background information is needed.

Parameters in the installation options data set

TRACKCLASSUSAGE(class1[,class2])

Indicates information about tracking key usage by classes of cryptographic operations. Reference date tracking must be enabled. See the KDSREFDAYS parameter description.

ICSF tracks the usage of keys in the common record format CKDS, PKDS, and TKDS. The usage is recorded in the metadata for the key record as the last date any service in a class was called. The reference period is the same as the reference date tracking. See the KDSREFDAYS parameter description.

The supported cryptographic classes are:

DATADEC

Symmetric keys data decryption operations.

When a symmetric key is referenced for these services, the date is recorded.

- Decipher (CSNBDEC, CSNEDEC, CSNBDEC1, and CSNEDEC1).
- Ciphertext Translate2 (CSNBCTT2, CSNECTT2, CSNBCTT3, and CSNECTT3):
  - Inbound key identifier.
- Symmetric Algorithm Decipher (CSNBSAD, CSNESAD, CSNBSAD1, and CSNESAD1).
- Symmetric Key Decipher (CSNBSYD, CSNESYD, CSNBSYD1, and CSNESYD1).
- Field Level Decipher (CSNBFLD and CSNEFLD).
- FPE Decipher (CSNBFPED and CSNEFPED).
- FPE Translate (CSNBFPET and CSNEFPET):
  - Inbound key identifier.
- Format Preserving Algorithms Decipher (CSNBFFXD and CSNEFFXD).
- Format Preserving Algorithms Translate (CSNBFFXT and CSNEFFXT).
  - Inbound key identifier.
- PKCS #11 Secret Key Decrypt (CSFPSKD and CSFPSKD6).
- PKCS #11 Secret Key Reencrypt (CSFPSKR and CSFPSKR6)
  - Inbound key identifier.

DATAENC

Symmetric keys data encryption operations.

When a symmetric key is referenced for these services, the date is recorded.

- Encipher (CSNBENC, CSNEENC, CSNBENC1, and CSNEENC1).
- Ciphertext Translate2 (CSNBCTT2, CSNECTT2, CSNBCTT3, and CSNECTT3):
  - Outbound key identifier.
- Symmetric Algorithm Encipher (CSNBSAE, CSNESAE, CSNBSAE1, and CSNESAE1).
- Symmetric Key Encipher (CSNBSYE, CSNESYE, CSNBSYE1, and CSNESYE1).
- Field Level Encipher (CSNBFLE and CSNEFLE).
- FPE Encipher (CSNBFPEE and CSNEFPEE).
- FPE Translate (CSNBFPET and CSNEFPET):
  - Outbound key identifier.
- Format Preserving Algorithms Encipher (CSNBFFXE and CSNEFFXE).
- Format Preserving Algorithms Translate (CSNBFFXT and CSNEFFXT):
  - Outbound key identifier.
- PKCS #11 Secret key encrypt (CSFPSKE and CSFPSKE6).
- PKCS #11 Secret Key Reencrypt (CSFPSKR and CSFPSKR6)
  - Outbound key identifier.

Migration

Callable services

For complete reference information on these callable services, see z/OS Cryptographic Services ICSF Application Programmer's Guide.

Note: When an APAR number is listed in the FMID column along with an ICSF FMID, the FMID is the earliest release where the new function is supported.

Table 9. Summary of new and changed ICSF callable services

Callable service	FMID	Description
Digital Signature Generate	HCR77D1 OA61253	Changed: Support for Schnorr digital signature algorithm.
Digital Signature Verify	HCR77D1 OA61253	Changed: Support for Schnorr digital signature algorithm.
Diversified Key Generate	HCR77D1 OA61253	Changed: Support for AusPayNet key derivation algorithms.
Diversified Key Generate2	HCR77D1 OA61253	Changed: Change derivation data length restrictions for KDFFM-DK. Add initialization vector support.
Diversify Directed Key	HCR77D1 OA61253	Changed: KTV changes for PINPROT and ISO-4
DK PIN Change	HCR77D1 OA61253	Changed: Support for General ISO PIN error mode. Allow specifying script MAC length via parameter.
DK PIN Verify	HCR77D1 OA61253	Changed: Support for General ISO PIN error mode
Encrypted PIN Translate	HCR77D1 OA61253	Changed: Support for General ISO PIN error mode
Encrypted PIN Translate2	HCR77D1 OA61253	Changed: Support for General ISO PIN error mode
Encrypted PIN Verify2	HCR77D1 OA61253	New: Verify a trial PIN against a reference PIN in an encrypted PIN block.

PKA Key Translate	HCR77D1 OA61253	Changed: Support for translating CCA PKA key token to Azure object format.
Random Number Generate Long	HCR77D1 OA61253	Changed: Support for encrypting the returned random number under a cipher key.
Symmetric Algorithm Decipher	HCR77D1 OA61253	Changed: Add support for X9.23 padding.
Symmetric Algorithm Encipher	HCR77D1 OA61253	Changed: Add support for X9.23 padding. Changed: Support for AusPayNet MAC generation and verification.
Symmetric Key Export	HCR77D1 OA61253	Changed: Support for translating CCA AES key token to Azure object format.

CCA access control

For complete reference information on these CCA access controls, see z/OS Cryptographic Services ICSF Application Programmer's Guide.

Note: When an APAR number is listed in the FMID column along with an ICSF FMID, the FMID is the earliest release where the new access control is supported

Table 10. Summary of new and changed CCA access controls

Access control	Description	FMID or APAR number	Services affected	Offset
General ISO PIN Error Mode	New	HCR77D1 OA61253	CSNBDPC CSNBDPV CSNBPTR CSNBPTR2	039F
Encrypted PIN Translate - Translate PIN Check Mode	New	HCR77D1 OA61253	CSNBPTR CSNBPTR2	03A0
Encrypted PIN Verify2 - REFPIN	New	HCR77D1 OA61253	CSNBPVR2	03B0
Encrypted PIN Verify2 - TRUNCPIN	New	HCR77D1 OA61253	CSNBPVR2	03B1
Symmetric Algorithm Encipher - allow A28MACGN and A28MACVR	New	HCR77D1 OA61253	CSNBSAE	03B2
Symmetric Algorithm Encipher - allow A28OWFCL	New	HCR77D1 OA61253	CSNBSAE	03B3
Symmetric Algorithm Encipher - allow A28OWFEC	New	HCR77D1 OA61253	CSNBSAE	03B4
Random Number Generate Long - TDES-CBC	New	HCR77D1 OA61253	CSNBRNGL	03B5
PKA Key Translate - From CCA RSA to CKM-RAKW format	New	HCR77D1 OA61253	CSNDPKT	03B6
PKA Key Translate - From CCA ECC to CKM-RAKW format	New	HCR77D1 OA61253	CSNDPKT	03B7
Symmetric Key Export - CKM-RAKW	New	HCR77D1 OA61253	CSNDSYX	03B8
Diversified Key Generate - A28OWFEC	New	HCR77D1 OA61253	CSNBDKG	03B9
Diversified Key Generate - A28OWFCL	New	HCR77D1 OA61253	CSNBDKG	03BA
Diversified Key Generate - A28XOREC	New	HCR77D1 OA61253	CSNBDKG	03BB

CICS attachment facility

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

Modify and include:

For FMID HCR77D1:

CSFFFXD, CSFFFXE, CSFFFXT (APAR OA59593).
CSFPVR2, CSFPSKR (APAR OA61253)

Resource Manager Interface (RMF)

Support to enable RMF to provide performance measurements on these selected ICSF services and functions. The measurements refer to these services processing on cryptographic coprocessors except for one-way hash. One-way hash is processed on CPACF.

- Decipher (CSNBDEC)
- Digital Signature Generate (CSNDDSG)
- Digital Signature Verify (CSNDDSV)
- Encipher (CSNBENC)
- Encrypted PIN Translate (CSNBPTR)
- Encrypted PIN Translate2 (CSNBPTR2)
- Encrypted PIN Translate Enhanced (CSNBPTRE)
- Encrypted PIN Verify (CSNBPVR)
- Encrypted PIN Verify2 (CSNBPVR2)
- Format Preserving Algorithms Decipher (CSNBFFXD)
- Format Preserving Algorithms Encipher (CSNBFFXE)
- Format Preserving Algorithms Translate (CSNBFFXT)
- FPE Decipher (CSNBPFED)
- FPE Encipher (CSNBPFEE)
- FPE Translate (CSNBPFET)
- MAC Generate (CSNBMGN)
- MAC Generate2 (CSNBMGN2)
- MAC Verify (CSNBMVR)
- MAC Verify2 (CSNBMVR2)
- One-Way Hash (CSNBOWH)
- Symmetric Algorithm Decipher (CSNBSAD)
- Symmetric Algorithm Encipher (CSNBSAE)

Diagnosis reference information

RMF measurements table

Table 116 on page 373 describes the contents of the performance measurements for RMF. The count fields are double-word length.

Table 11. RMF measurements record format			
Offset (Dec)	**Number of Bytes**	**Field name**	**Description**
272	8	DACC_ENT_ID	Identifier of count array - character PVR. The PIN Verify and PIN Verify2 services will collect data as follows: • Collect the number of service calls only.
280	8	DACC_ENT_SVC_CNT	Count of PVR service calls.

ICSF SMF records

Table 12. Subtype 48 Compliance warning event					
Dec	**Hex**	**Name**	**Length**	**Format**	**Description**
264	108	SMF82_TAG_TOK_FMT	1	binary	The format of the token. X'01' Fixed length CCA token. X'02' Variable length CCA token. X'03' TR-31 key block. X'04' RKX token. X'05' RSA DSI PKCS #1 V2 OAEP format (PKCSOAEP). X'06' RSA DSI PKCS #1 block type 02 format (PKCS-1.2). X'07' Zero padded (ZERO-PAD). X'08' PKA92 format (PKA92). X'09' EMV or Smart Card format (EMVCRT, EMVDDA, EMVDDAE, SCCOMCRT, SCCOMME, or SCVISA). X'0A' Azure key object (CKM_RAKW) **Notes**: 1. When format is X'04' or greater, no other key or token-related fields are present. 2. When format is TR-31 key block, the only other key or token related field that can be present is the key fingerprint.

Resource names for CCA and ICSF entry points

Table13. Resource names for CCA and ICSF entry points						
Descriptive service name	**CCA entry point name**		**ICSF entry point name**		**SAF resource name**	**Callable service exit name**
Encrypted PIN Verfiy2	CSNBPVR2	CSNBEPVR2	CSFPVR2	CSFPVR26	CSFPVR2	CSFPVR2
PKCS #11 Secret Key Reencrypt	N/A	N/A	CSFPSKR	CSFPSKR6	CSF1SKR[1]	N/A

Notes:

– [1] CSF1xxx is just another name for the CSFPxxx service.

Chapter 5. Update of z/OS Cryptographic Services ICSF Application Programmer's Guide, SC14-7508-10, information

This topic contains updates to the document z/OS Cryptographic Services ICSF Application Programmer's Guide, SC14-7508-09, for the updates provided by this APAR. Refer to this source document if background information is needed.

Managing personal authentication

Verifying credit card data

Encrypted PIN Verify2 Callable Service (CSNBPVR2 and CSNEPVR2)

To verify a supplied PIN against a reference PIN, call the Encrypted PIN verify2 callable service. You need to specify the supplied enciphered PIN block, the reference enciphered PIN block, the PIN-encrypting keys that encipher the blocks, and other relevant data. The service compares the two personal identification numbers; if they are the same, it verifies the supplied PIN. IBM 3624, ISO-0, ISO-1, ISO-2, ISO-3, and ISO-4 PIN block formats are supported. See Chapter 8, "Financial services," on page 667 for additional information.

An enhanced PIN security mode is available for extracting PINs from encrypted PIN blocks. This mode only applies when specifying a PIN-extraction method for an IBM 3621 or an IBM 3624 PIN-block. See "Encrypted PIN Verify2 (CSNBPVR and CSNEPVR2)" on page XXX for more information.

Australian Payment Network support

This topic describes the support for the Australian Payment Network that based on standard AS2805.5.4.

Key derivation

CSNBDKG supports key derivation to meet the needs of the APN.
CSNBRNGL supports encrypting the output under a data-encrypting key.

MAC generation

CSNBSAE supports generating and verifying MACs and related processing.

Summary of callable services

Table 14. Summary of ICSF callable services		
Service	**Service name**	**Function**
Chapter 8, “Financial services,” on page 667		
CSNBPVR2 CSNEPVR2	Encrypted PIN Verify2	Verifies a supplied PIN against a reference PIN. DUKPT keywords are supported.

Summary of the PKA callable services

Table 15. Summary of PKA callable services		
Service	**Service Name**	**Function**
Chapter 12, “Managing PKA cryptographic keys,” on page 1059		
CSNDPKT CSNFPKT	PKA key translate	Translates a CCA RSA key token to a smart card format or a PKCS #11 object. Convert an CCA RSA key token with a DES OPK to a token with an AES OPK. Convert CCA PKA key token to a compliant-tagged token.

PKCS #11 services

PKCS #11 tokens and objects

ICSF provides callable services that support PKCS #11 token and object creation and use. The following table summarizes these callable services. For complete syntax and reference information, refer to Chapter 16, “Using PKCS #11 tokens and objects,” on page 1183

Table 16. Summary of PKCS #11 callable services		
Verb	**Service Name**	**Function**
PKCS #11 Secret Key Reencrypt	CSFPSKR	Decrypts and re-encrypts data using secure secret keys

Diversified Key Generate (CSNBDKG and CSNEDKG)

Use the Diversified Key Generate service to generate a key based on the key-generating key, the processing method, and the parameter supplied. The control vector of the key-generating key also determines the type of target key that can be generated.

To use this service, specify:

- The rule array keyword to select the diversification process.
- The operational key-generating key from which the diversified keys are generated. The control vector associated with this key restricts the use of this key to the key generation process. This control vector also restricts the type of key that can be generated.
- The data and length of data used in the diversification process.
- The generated-key may be an internal token or a skeleton token containing the desired CV of the generated-key. The generated key CV must be one that is permitted by the processing method and the key-generating key. The generated-key will be returned in this parameter.
- A key generation method keyword.

This service generates diversified keys as follows:

- Determines if it can support the process specified in rule array.
- Recovers the key-generating key and checks the key-generating key class and the specified usage of the key-generating key.
- Determines that the control vector in the generated-key token is permissible for the specified processing method.
- Determines that the control vector in the generated-key token is permissible by the control vector of the key-generating key.
- Determines the required data length from the processing method and the generated-key CV. Validates the data_length.
- Generates the key appropriate to the specific processing method. Adjusts parity of the key to odd.

Creates the internal token and returns the generated diversified key.

The callable service name for AMODE(64) invocation is CSNEDKG.

Format

```
CALL CSNBDKG(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              generating_key_identifier,
              data_length,
              data,
              data_decrypting_key_identifier,
              generated_key_identifier)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems.Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The only valid value is 1, 2, or 3.

rule_array

Direction	Type
Input/Output	String

Keywords that provide control information to the callable service. The processing method is the algorithm used to create the generated key. The keywords must be 8 bytes of contiguous storage with the keyword left-justified in its 8-byte location and padded on the right with blanks.

Table 17. Rule Array Keywords for Diversified Key Generate

Keyword	Meaning
Processing Method for generating or updating diversified keys (required)	
A28OWFCL	Specifies that 16 bytes of clear data will be processed as described in AS2805.5.4 to create the generated key. The data parameter will be processed by the AusPayNet One Way Function using a double-length key-encrypting key to generate a new key-encrypting key.

A28OWFEC	Specifies that 16 bytes of data encrypted using the data_decrypting_key_identifier will be processed as described in AS2805.5.4 to create the generated key. The data parameter will be processed by the AusPayNet One Way Function using a double-length key-encrypting key to generate a new key-encrypting key. The data parameter should contain the data wrapped as follows: data = wrap(PPASN) || wrap(PPASN)
A28XORE	Specifies that 16 bytes of data encrypted using the data_decrypting_key_identifier will be processed as specified in AS2805.5.4 to generate a key-encrypting key.
CLR8-ENC	Specifies that 8-bytes of clear data shall be multiply-encrypted with the generating key. The generating_key_identifier must be a KEYGENKY key type with bit 19 of the control vector set to 1. The control vector in generated_key_identifier must specify a single-length key. The key type may be DATA, MAC, or MACVER. Note: CIPHER class keys are not supported.
SESS-XOR	Modifies an existing DATA, DATAC, MAC, DATAM, MACVER, or DATAMV single-length or double-length key. Specifies the VISA method for session key generation. Data supplied may be 8 or 16 bytes of data depending on whether the generating_key_identifier is a single or double length key. The 8 or 16 bytes of data is XORed with the clear value of the generating_key_identifier. The generated_key_identifier has the same control vector as the generating_key_identifier. The generating_key_identifier may be DATA/DATAC, MAC/DATAM or MACVER/DATAMV key types.
TDES-DEC	Data supplied may be 8 or 16 bytes of clear data. If the generated_key_identifier specifies a single length key, then 8-bytes of data is TDES decrypted under the generating_key_identifier. If the generated_key_identifier specifies a double length key, then 16-bytes of data is TDES ECB mode decrypted under the generating_key_identifier. No formating of data is done prior to encryption. The generating_key_identifier must be a DKYGENKY key type, with appropriate usage bits for the desired generated key.
TDES-ENC	Data supplied may be 8 or 16 bytes of clear data. If the generated_key_identifier specifies a single length key, then 8-bytes of data is TDES encrypted under the generating_key_identifier. If the generated_key_identifier specifies a double length key, then 16-bytes of data is TDES ECB mode encrypted under the generating_key_identifier. No formatting of data is done prior to encryption. The generating_key_identifier must be a DKYGENKY key type, with appropriate usage bits for the desired generated key. The generated_key_identifier may be a single or double length key with a CV that is permitted by the generating_key_identifier.
TDES-CBC	Data supplied must be 16 bytes of clear data. The generated_key_identifier must specify a double length key, then the 16 bytes of data is TDES-CBC mode encrypted under the generating_key_identifier. No formatting of data is done prior to encryption. The generating_key_identifier must be a DKYGENKY key type, with appropriate usage bits for the desired generated key. The generated_key_identifier must be a double length key with a CV that is permitted by the generating_key_identifier.
TDES-XOR	Combines the function of the existing TDES-ENC and SESS-XOR into one step.

	The generating key must be a level 0 DKYGENKY and cannot have replicated halves. The session key generated must be double length and the allowed key types are DATA, DATAC, MAC, MACVER, SMPIN and SMKEY. Key type must be allowed by the generating key control vector.
TDESEMV2	Supports generation of a session key by the EMV 2000 algorithm (This EMV2000 algorithm uses a branch factor of 2). The generating key must be a level 0 DKYGENKY and cannot have replicated halves. The session key generated must be double length and the allowed key types are DATA, DATAC, MAC, MACVER, SMPIN and SMKEY. Key type must be allowed by the generating key control vector.
TDESEMV4	Supports generation of a session key by the EMV 2000 algorithm (This EMV2000 algorithm uses a branch factor of 4). The generating key must be a level 0 DKYGENKY and cannot have replicated halves. The session key generated must be double length and the allowed key types are DATA, DATAC, MAC, MACVER, SMPIN and SMKEY. Key type must be allowed by the generating key control vector.
Key Wrapping Method (optional)	
USECONFG	Specifies that the system default configuration should be used to determine the wrapping method. This is the default keyword. The system default key wrapping method can be specified using the DEFAULTWRAP parameter in the installation options data set. See the z/OS Cryptographic Services ICSF System Programmer's Guide.
WRAP-ENH	Use enhanced key wrapping method, which is compliant with the ANSI X9.24 standard.
WRAPENH3	Specifies to wrap the key using the enhanced wrapping method with SHA-256 and CMAC authentication code.
WRAP-ECB	Use original key wrapping method, which uses ECB wrapping for DES key tokens and CBC wrapping for AES key tokens.
Translation Control (optional)	
ENH-ONLY	Restrict rewrapping of the key_identifier token. Once the token has been wrapped with the enhanced method, it cannot be rewrapped using the original method. This is the default when the wrapping method is WRAPENH3.

generating_key_identifier

Direction	**Type**
Input/Output	String

The identifier of the key-generating key. The key identifier is a 64 byte operational token or the key label of an operational token in key storage. The type of key depends on the processing method.

The key is a DES key-generating key as described in the rule array.

For A28XOREC, this key must be a double-length DES EXPORTER key.
For A28OWFEC, this key must be a double-length DES EXPORTER or DES CIPHER key.
For A28OWFCL, this key must be a double-length DES CIPHER key.

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

data_length

Direction	Type
Input	Integer

The length of the data parameter in bytes. The required length depends on the diversification process specified in the rule array and the length of the key identified by the generated_key_identifier parameter:

Rule-array keyword	Key length of generated key	Required data length
CLR8-ENC	SINGLE	8
A28OWFCL, A28OWFEC, A28XOREC	DOUBLE	16
TDES-CBC	DOUBLE or null key-token	16
TDES-ENC	DOUBLE or null key-token	16
	SINGLE	8
TDES-DEC	DOUBLE or null key-token	16
	SINGLE	8
TDESEMV2, TDESEMV4	DOUBLE	10, 18, 26, or 34
TDES-XOR	DOUBLE	10 or 18
SESS-XOR	DOUBLE	16
	SINGLE	8

data

Direction	Type
Input	String

Data input to the diversified key or session key generation process. Data depends on the processing method and the generated_key_identifier.

For TDESEMV4 or TDESEMV2, the data is either 18 bytes (36 digits) or 34 bytes 68 digits) of data comprised of:

- 16 bytes (32 digits) of card specific data used to create the card specific intermediate key (UDK) as per the TDES-ENC method. This will typically be the PAN and PAN Sequence number as per the EMV specifications
- 2 bytes (4 digits) of ATC (Application Transaction Count)
- (optional) 16 bytes (32 digits) of IV (Initial Value) used in the EMV

data_decrypting_key_identifier

Direction	Type
Input/Output	String

The key to decrypt the value supplied in the data parameter. When the processing method rule is A28OWFEC or A28XOREC, this parameter must contain the label or 64-byte key token of a DES CIPHER or DECIPHER key. The key must be a double-length key.

Otherwise, this parameter must contain a 64-byte null token.

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

generated_key_identifier

Direction	Type
Input/Output	String

The key to be generated. On input, specify a null token or an internal token or a skeleton token containing the control vector of the key to be generated. On output, this parameter contains the generated key.

- For the CLR8-ENC, TDESEMV2, TDESEMV4, and TDES-XOR keywords, a null token must not be specified.
- For the TDES-CBC, TDES-ENC, or TDES-DEC keywords, either a null key- token or an internal key-token must be specified.
- For the SESS-XOR and keyword, a null key-token must be specified.
- For the A28XOREC keyword, a DES EXPORTER key token or skeleton must be specified.
- For the A28OWFEC keyword, a DES EXPORTER or DES CIPHER key token or skeleton must be specified. The supplied token must match the key supplied in the generating_key_identifier parameter.
- For the A28OWFCL keyword, a DES CIPHER, DES MAC with sub-type ANY-MAC, or DES IPINENC key token or skeleton must be specified.

To generate a compliant-tagged key token, a compliant-tagged skeleton token must be supplied.

When the WRAPENH3 method is selected, a skeleton key token is required. A secure internal key token wrapped with the WRAPENH3 method obfuscates the key length.

The output generated_key_token will use the default wrapping method unless a rule array keyword overriding the default is specified.

The DES key wrapping methods available are described in “Key wrapping”.

Restrictions

This callable service does not support version X'10' external DES key tokens (RKX key tokens).

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

Refer to Appendix C, “Control vectors and changing control vectors with the CVT callable service,” on page 1429 for information on the control vector bits for the DKG key generating key.

For Session key algorithm (EMV Smartcard specific), a master derivation key (MDK) can be used in two ways:

- To calculate the Card Specific Key (or UDK) in the personalization process, call this service with the TDES-ENC or TDES-CBC method using an output token that has been primed with the CV of the final session key, for instance, if the MDK is a DMPIN, the token should have the CV of an SMPIN key; DMAC (a double length MAC); DDATA (a double length DATA key), and so on.

  The result would then be exported in the personalization file. This key is not usable in this form for any other calculations.

- To use the session key, call this service with the TDESEMV4 method. Provide, for input, the same card data that was used to create the UDK as well as the ATC and optionally the IV value. This is the key that will be used in EMV related Smartcard processing.

  This same processing applies to those API's the generate the session key on your behalf, like CSNBPCU.

If ICSF is configured to audit the lifecycle of tokens [AUDITKEYLIFECKDS(TOKEN(YES),...) is specified], an additional request is made to the Crypto Express coprocessor to generate the key fingerprint to be used for auditing the generated key.

Access control points

The following table shows the access control points in the domain role that control the function of this service.

Table 18. Required access control points for Diversified Key Generate

Rule array keyword	Access control point
CLR8-ENC	Diversified Key Generate - CLR8-ENC
A28XOREC	Diversified Key Generate - A28XOREC
A28OWFCL	Diversified Key Generate - A28OWFCL
A28OWFEC	Diversified Key Generate - A28OWFEC
SESS-XOR	Diversified Key Generate - SESS-XOR
TDES-DEC	Diversified Key Generate - TDES-DEC
TDES-ENC	Diversified Key Generate - TDES-ENC
TDES-CBC	Diversified Key Generate - TDES-CBC
TDES-XOR	Diversified Key Generate - TDES-XOR
TDESEMV2 or TDESEMV4	Diversified Key Generate - TDESEMV2/TDESEMV4

When the key wrapping method keyword specifies a wrapping method that is not the default method, the **Diversified Key Generate - Allow wrapping override keywords** access control must be enabled.

When a key-generating key of key type DKYGENKY is specified with control vector bits (19 – 22) of B'1111', the **Diversified Key Generate - DKYGENKY – DALL** access control point must also be enabled in the domain role.

When using the TDES-ENC or TDES-DEC modes, you can specifically enable generation of a single-length key or a double-length key with equal key-halves by enabling the **Diversified Key Generate – Single length or same halves** access control point.

When the **Disallow 24-byte DATA wrapped with 16-byte Key** access control point is enabled, this service will fail if the source key is a triple-length DATA key and the DES master key is a 16-byte key.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 19. Diversified Key Generate required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	Keywords ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, TDES-CBC, A28OWCEC, A28OWFCL, and A28XOREC not supported. Enhanced key token wrapping not supported. Compliant-tagged key tokens are not supported. Rule array keyword WRAPENH3 is not supported
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	Keywords ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, TDES-CBC, A28OWCEC, A28OWFCL, and A28XOREC not supported. Enhanced key token wrapping not supported. Compliant-tagged key tokens are not supported. Rule array keyword WRAPENH3 is not supported
	Crypto Express3 Coprocessor	Keywords TDES-CBC, A28OWCEC, A28OWFCL, and A28XOREC are not supported. Enhanced key token wrapping not supported. Compliant-tagged key tokens are not supported. Rule array keyword WRAPENH3 is not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	Keywords TDES-CBC, A28OWCEC, A28OWFCL, and A28XOREC are not supported. Compliant-tagged key tokens are not supported. Rule array keyword WRAPENH3 is not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	TDES-CBC support requires the Sep. 2013 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported.

		Rule array keywords WRAPENH3, A28OWCEC, A28OWFCL, and A28XOREC are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. Rule array keyword WRAPENH3 requires the April 2021 or later licensed internal code (LIC). Rule array keywords A28OWCEC, A28OWFCL, and A28XOREC are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. Rule array keyword WRAPENH3 requires the April 2021 or later licensed internal code (LIC). Rule array keywords A28OWCEC, A28OWFCL, and A28XOREC are not supported.
	Crypto Express6 CCA Coprocessor	Rule array keyword WRAPENH3 requires the April 2021 or later licensed internal code (LIC). Rule array keywords A28OWCEC, A28OWFCL, and A28XOREC are not supported.
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. Rule array keyword WRAPENH3 requires the April 2021 or later licensed internal code (LIC). Rule array keywords A28OWCEC, A28OWFCL, and A28XOREC are not supported.
	Crypto Express6 CCA Coprocessor	Rule array keyword WRAPENH3 requires the April 2021 or later licensed internal code (LIC). Rule array keywords A28OWCEC, A28OWFCL, and A28XOREC are not supported.
	Crypto Express7 CCA Coprocessor	Rule array keyword WRAPENH3 requires the April 2021 or later licensed internal code (LIC). Rule array keywords A28OWCEC, A28OWFCL, and A28XOREC require the CCA release 7.4 or later licensed internal code.

Diversified Key Generate2 (CSNBDKG2 and CSNEDKG2)

The Diversified Key Generate2 callable service generates an AES key based on a function of a key-generating key, the process rule, and data that you supply.

To use this service, specify:

- The rule array keyword to select the diversification process.
- The operational AES key-generating key from which the diversified keys are generated.

  For a key-generating key with a key-derivation sequence level of 1 or 2:

  The type of key created will be a DKYGENKY key with a sequence level one lower than the key-generating key with the same key usage fields.

  For a key-generating key with a key-derivation sequence level of 0:

  Key usage field 1 determines the type of key that is generated and restricts the use of this key to the key-diversification process. If the generating key has related key usage fields 3 through field 6 defined, these key usage attributes are used to control the permitted key usage attributes for the key to be generated.

  Note: Key usage field 2 of the generating DKYGENKY key contains a flag in its high-order byte. This flag byte determines how key usage fields 3 and beyond (called the related generated key usage fields) are used to control the values of the key usage fields of the generated key:

  – When the type of key to diversify is D-ALL, the flag is undefined because there are no key usage restrictions on the generated key. The generating key has no related generated key usage fields.

  – When the type of key to diversify is not D-ALL and the flag byte has KUF-MBE usage, the key usage fields of the key to be generated must be equal to the related generated key usage fields that start with key usage field 3 of the generating key.

  – When the type of key to diversify is not D-ALL and the flag byte has KUF-MBP usage, the key usage fields of the key to be generated must be permissible. In other words, a key to be diversified is only permitted to have a level of usage less than or equal to the related key usage fields (key usage fields starting with key usage field 3). One exception is that the UDX-only setting of the generated key always must be equal to the UDX-ONLY setting of the generating key.

- The diversification data and length of data used in the diversification process.
- The variable-length AES symmetric-key generated token with a suitable key type and key usage fields for receiving the diversified key, or a null key token if the type of key to diversify supports default key usage and a default key is desired.

The callable service name for AMODE(64) invocation is CSNEDKG2.

Format

```
CALL CSNBDKG2(
```

```
return_code,
reason_code,
exit_data_length,
exit_data,
rule_array_count,
rule_array,
generating_key_identifier_length,
generating_key_identifier,
derivation_data_length,
derivation_data,
input_initial_vector_length,
input_initial_vector,
reserved2_length,
reserved2,
generated_key_identifier1_length,
generated_key_identifier1,
generated_key_identifier2_length,
generated_key_identifier2)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value must be 1 or 2.

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 20. Rule array keywords for Diversified Key Generate2

Keyword	Meaning
Diversification Process (required)	
KDFFM-DK	Specifies to use the DK version of key derivation function in feedback mode. This method uses AES CMAC to encipher the ~~16 to 40 bytes of~~ derivation data with the k-bit diversified key generating key (banking association specific master key) to produce a k-bit generated bank specific Issuer Master Key, where k = 128, 192, or 256.
MK-OPTC	Specifies to use the EMV master key derivation option C specified in EMV Integrated Circuit Card Specifications for Payments Systems. This method uses AES in ECB mode to encipher the 16 bytes of derivation data with the k-bit diversified key generating key (Issuer Master Key) to produce a k-bit generated ICC master key, where k = 128, 192, or 256.
SESS-ENC	Specifies to use the EMV common session key derivation option specified in EMV Integrated Circuit Card Specifications for Payments Systems. This method uses AES in ECB mode to encipher the 16 bytes of derivation data with the k-bit diversified key generating key (ICC master key) to produce a k-bit generated key (ICC session key), where k = 128, 192, or 256.
Bit length of generated key (one, optional). Valid only with the KDFFM-DK keyword. Default is to use the bit length of the generating key as the bit length of the generated key.	
KLEN128	Specifies the bit length of the generated key to be 128.
KLEN192	Specifies the bit length of the generated key to be 192, allowed if and only if the bit length of the enerating key is greater than or equal to 192.
KLEN256	Specifies the bit length of the generated key to be 256, allowed if and only if the bit length of the generating key is 256.
IV Usage (One optional) Valid only with process keyword KDFFM-DK.	
DEFLT-IV	Specifies to use the DK default initial vector value as the IV in the derivation function. This is the default value.
USE-IV	Specifies to use the value specified in the input_initial_vector parameter as the IV in the derivation function.

generating_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the generating_key_identifier parameter. If the generating_key_identifier contains a label, the value must be 64. Otherwise, the value must be between the actual length of the token and 725.

generating_key_identifier

Direction	Type
Input/Output	String

The identifier of the key-generating key. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be AES and the key type must be DKYGENKY. The key usage field indicates the key type of the generated key. The key length determines the length of the generated key.

If SESS-ENC is specified, the clear length of the generated key is equal to the clear length of the generating key. If SESS-ENC is specified, the key-derivation sequence level must be set to DKYL0 in the key usage field 2.

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

derivation_data_length

Direction	Type
Input	Integer

Specifies the length in bytes of the derivation_data parameter. If SESS-ENC or MK-OPTC is specified, the value must be 16.

When the process rule KDFFM-DK is specified, the value must be between 1 to 2048 inclusive for CCA release 6.7, 7.4, and later. Otherwise, the value must be 16 to 40 inclusive.

derivation_data

Direction	Type
Input	String

The derivation data to be used in the key generation process. This data is often referred to as the diversification data. For SESS-ENC, the derivation data is 16-bytes long.

Note that if SESS-ENC is specified and the length of the key generating key is 192 bits or 256 bits, the data is manipulated in conformance with the EMV Common Session Key Derivation Option.

input_initial_vector_length

Direction	Type
Input	Integer

Length in bytes of the input_initial_vector parameter. For CCA releases 6.7, 7.4 and later when the KDFFM-DK process rule and the USE-IV keywords are specified, the value must be between 0 or 16 inclusive. Otherwise, the value must be 0.

input_initial_vector

Direction	Type
Input	String

For the KDFFM-DK process rule, the 16-byte initial vector value for the algorithm When a value is not provided, the default value, 0x52525252525252522525252525252525, will be used. When the USE-IV keyword is specified and the input_initial_vector_length is 0, the initial value will be hex zero.

When the input_initial_vector_length is zero, this field is ignored.

reserved2_length

Direction	Type
Input	Integer

Length in bytes of the reserved2 parameter. The value must be 0.

reserved2

Direction	Type
Input	String

This field is ignored.

generated_key_identifier1_length

Direction	Type
Input/Output	Integer

On input, the length of the buffer for the generated_key_identifier1 parameter in bytes. The maximum value is 725 bytes.

On output, the parameter holds the actual length of the generated_key_identifier1 parameter.

generated_key_identifier1

Direction	Type
Input/Output	String

The buffer for the generated key token.

On input, the buffer contains a null token or a valid internal skeleton token containing the desired key-usage fields and key-management fields you want to generate. The key token must be left justified in the buffer.

The generating key (generating_key_identifier parameter) determines whether on input the generated_key_identifier1 parameter can identify a null key token or a skeleton key token.

When the generating_key_identifier is compliant-tagged, a compliant-tagged key token will be created.

When a skeleton token is passed as input and the generating_key_identifier is compliant-tagged, the skeleton token must have the compliant-tagged flag on.

Table 21. Summary of input generating key tokens, input generated key tokens, and output generated key tokens

Input generating key token	Input generated key token	Output generated key token
DKYL0, type of key to diversify D-ALL	Skeleton key token required.	Key type same as skeleton, diversified key final.
DKYL0, type of key to diversify not D-ALL	Null or skeleton key token allowed.	Key type determined by input generated key token type of key to diversify. If null key token on input, the output key token will have attributes based on the related generated key usage fields of the input generating key token. Otherwise, the output key token will have attributes of input skeleton key token.
DKYL1, any type of key to diversify	Null key token required.	Same as input generating key token except DKYL0 and with new level of diversified key.
DKYL2, any type of key to diversify	Null key token required.	Same as input generating key token except DKYL1 and with new level of diversified key.

Notes:

1. If the supplied generated key-token contains a key, the key value and length are ignored and overwritten.
2. If the generating_key_identifier1 parameter identifies a DKYGENKY key token with a key-derivation sequence level of DKYL0 and it does not have a type of key to diversify of D-ALL, the key type must match what the generating key indicates can be created in the key generating key usage field at offset 45.
3. The key usage fields in the generated key must meet the requirements (KUF 'must be equal' or 'must be permitted') of the corresponding key usage fields in the generating key unless D-ALL is specified in the generating key. A flag bit in the DKYGENKY key-usage field 2 determines whether the key-usage field level of control is KUF-MBE or KUF-MBP.
4. If authorized by access control, D-ALL permits the derivation of several different keys. On output, the buffer contains the generated key token.

generated_key_identifier2_length

Direction	Type
Input/Output	Integer

Length in bytes of the generated_key_identifier2 parameter. The value must be 0.

generated_key_identifier2

Direction	Type
Input/Output	String

This field is ignored.

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS.

If ICSF is configured to audit the lifecycle of tokens [AUDITKEYLIFECKDS(TOKEN(YES),...) is specified], an additional request is made to the Crypto Express coprocessor to generate the key fingerprint to be used for auditing the generated key.

Access control points

The following table shows the access control points in the domain role that control the function of this service:

Table 22. Required access control points for Diversified Key Generate2

Rule array keyword	Access control point
KDFFM-DK	Diversified Key Generate2 - KDFFM-DK
MK-OPTC	Diversified Key Generate2 - MK-OPTC
SESS-ENC	Diversified Key Generate2 - SESS-ENC

To use the KLEN192 and KLEN256 keywords, **the Diversified Key Generate2 - Allow length option with KDFFM-DK** access control point must be enabled.

If the key-generating key key-usage fields indicate that all key types may be derived, the **Diversified Key Generate2 – DALL** access control point must be enabled in the domain role.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 23. Diversified Key Generate2 required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC		This service is not supported.
IBM System z10 EC IBM System z10 BC		This service is not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	Requires the November 2013 or later licensed internal code (LIC). Keywords KDFFM-DK, MK-OPTC, KLEN128, KLEN192, and KLEN256 are not supported. Compliant-tagged key tokens are not supported. The input_initial_vector_length parameter value must be 0. The derivation_data_length must not exceed 40.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Requires the September 2013 or later licensed internal code (LIC). Keywords KDFFM-DK, MK-OPTC, KLEN128,

		KLEN192, and KLEN256 require the June 2015 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. The input_initial_vector_length parameter value must be 0. The derivation_data_length must not exceed 40.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Keywords KDFFM-DK, MK-OPTC, KLEN128, KLEN192, and KLEN256 require the June 2015 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. The input_initial_vector_length parameter value must be 0. The derivation_data_length must not exceed 40.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Keywords KDFFM-DK, MK-OPTC, KLEN128, KLEN192, and KLEN256 require the June 2015 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. The input_initial_vector_length parameter value must be 0. The derivation_data_length must not exceed 40.
	Crypto Express6 CCA Coprocessor	Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC). The input_initial_vector_length and derivation_data_length parameter support requires the require CCA release 6.7 or later licensed internal code.
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. The input_initial_vector_length parameter value must be 0. The derivation_data_length must not exceed 40.
	Crypto Express6 CCA Coprocessor	The input_initial_vector_length and derivation_data_length parameter support requires the require CCA release 6.7 or later licensed internal code.
	Crypto Express7 CCA Coprocessor	The input_initial_vector_length and derivation_data_length parameter support requires the CCA release 7.4 or later licensed internal code.

Diversify Directed Key (CSNBDDK and CSNEDDK)

The Diversify Directed Key callable service is used to selectively generate and derive a pair of associated keys in connection with a directed key diversification key scheme. The objective of the concept is to generate and derive a key pair with different key usages from one key diversification key (KDK). Key direction comes into play in that one of the keys is generated and is used for one direction (for example, encryption, MAC generate, and so forth), while the other key is derived and will have usage associated with a different direction (for example, decryption, MAC verification, and so forth). This callable service provides an option to perform the generate or derive operation.

A structure called a key type vector, which is always used as the initialization vector for the diversification process, is passed in as input and is used to determine what and how the key is produced by this callable service.

The key generated by this callable service is used as a session key. The intention in this context is that the keys of a generated and derived key pair are one-time keys. The key management fields of the output key will indicate that the key cannot be exported.

The callable service name for AMODE(64) invocation is CSNEDDK.

Format

```
CALL CSNBDDK(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              kdk_key_identifier_length,
              kdk_key_identifier,
              key_type_vector_length,
              key_type_vector,
              additional_derivation_data_length,
              additional_derivation_data,
              random_data_length,
              random_data,
              output_key_identifier_length,
              output_key_identifier)
```

Parameters

return_code

Direction	Type
Input/Output	String

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the return codes.

reason_code

Direction	Type
Input/Output	String

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value must be 2.

rule_array

Direction	Type
Input	Character

The rule_array contains keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 24. Keywords for Diversify Directed Key

Keyword	Meaning
Diversification Process (One required)	
KDFFM	Specifies to use the Key Derivation Function (KDF) in Feedback Mode (NIST SP 800-108) to generate key. The key type vector is used as the IV for this process.
Function (one required)	
DERIVE	Specifies to derive the passive diversified key of a pair of directed keys.
GENERATE	Specifies to generate the active diversified key of a pair of directed keys.

kdk_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the kdk_key_identifier parameter. If the kdk_key_identifier contains a label, the value must be 64. Otherwise, the value must be between the actual length of the token and 725.

kdk_key_identifier

Direction	Type
Input/Output	String

The identifier of the key diversification key used to derive keys. The key identifier is an operational token or the key label of an operational token in key storage.

The key algorithm of this key must be AES and the key type must be KDKGENKY. The key usage fields indicate the type of key to diversify and if the key is to be derived for entity A or entity B.

Note: When the GENERATE function is specified and the generating key has usage of KDKTYPEA, the associated DERIVE function must have usage of KDKTYPEB. Likewise, when the GENERATE function is specified and this key has usage of KDKTYPEB, the associated DERIVE function must have usage of KDKTYPEA.

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

key_type_vector_length

Direction	Type
Input	Integer

Specifies the length in bytes of the key_type_vector parameter. The value must be 16.

key_type_vector

Direction	Type
Input	String

The 16-byte key_type_vector specifies the rules for the calculation of the key value to be generated or derived and contains information needed to restrict the usage of the key to be generated or derived. The format of the structure is as follows:

Offset	Length	Description
0	2	Version number X'0000'.
2	2	Type of key to be derived or generated. **Value** **Meaning** X'0000' MAC. X'0001' Data encryption (cipher). X'0003' PIN encryption. X'0004'

		Key wrapping. All other values are reserved and undefined.
4	2	Key algorithm. **Value** **Meaning** X'0002' AES. All other values are reserved and undefined.
6	2	Length of the key to be derived or generated in bits. **Value** **Meaning** X'0800' 2048 (for example, AES-256).
8	2	Key usage restriction 1 of the key to be derived or generated, based on the key type field (value at offset 2): **For MAC key type** (value at offset 2 = X'0000'). **Value** **Meaning** X'0001' Key can derive or generate a CMAC mode key only. All other values are reserved and undefined. **For data encryption** (cipher) key type (value at offset 2 = X'0001'). **Value** **Meaning** X'0002' Key can derive or generate a CBC mode key only. All other values are reserved and undefined. **For PIN encryption key type** (value at offset 2 = X'0003'). **Value** **Meaning** X'0000' or X'0002' Key can derive or generate an ISO-4 format key only. See the note for KTVs for this key type. All other values are reserved and undefined. **For Key wrap key type** (value at offset 2 = X'0004'). **Value** **Meaning** X'0001' Key can derive or generate a VARDRV-D key only. All other values are reserved and undefined. **For all other key types not listed above:** **Value** **Meaning** X'0000' No key usage restriction 1. All other values are reserved and undefined.
10	2	**Key usage restriction 2** of the key to be derived or generated, depending on the key type (offset 2) and key usage restriction 1 (offset 8): **MAC key type and HMAC mode** (value at offset 2 = X'0000' and

		offset 8 = X’0001’). **Value** **Meaning** X'0002' SHA-256. X'0003' SHA-384. X'0004' SHA-512. All other values are reserved and undefined. **Key-wrap key type and VARDRV-D mode** (value at offset 2 = X’0004’ and offset 8 = X’0001’). **Value** **Meaning** X'0100' Maximum key length of the protected keys is 2048 bits. All other values are reserved and undefined. **All other values** at offset 2 and offset 8. **Value** **Meaning** X'0000' Undefined. All other values are reserved and undefined. **For all other key types and key usage restriction 1 combinations not listed above**: **Value** **Meaning** X'0000' No key usage restriction 2. All other values are reserved and undefined.
12	3	Reserved, must be binary zero.
15	1	**Key direction variant indicator** Diversifies the key to be derived or generated depending on the permitted use of direction. The HSM has to restrict the usage of the key depending on this value and the type of entity (A or B) which is an additional parameter in the process of deriving or generating the key. This value affects a key usage attribute of the key to be derived or generated. **Value** **Meaning** X'00' A ↔ B (undirected use of key). X'01' A → B (A active, B passive use of key). X'10' A ← B or equivalent (A passive, B active use of key). X'FF' System is to determine key direction from entity usage of the KDKGENKY and rule array keywords.

		KDK-A + GENERATE rule array keyword Direction set to X'01' (A → B). **KDK-B + GENERATE rule array keyword** Direction set to X'10' (A ← B). **KDK-A + DERIVE rule array keyword** Direction set to X'10' (A ← B). **KDK-B + DERIVE rule array keyword** Direction set to X'01' (A → B). All other values are reserved and undefined.

The following tables define the valid KTV supported.

For each of the four key types defined at KTV offset 2 (MAC, data encryption, PIN encryption, and key wrapping), there are two KTVs defined, with the only difference between them being the key direction variant indicator (KTV offset 15). Either entity Type A is active and Type B is passive (A→B), or Type B is active and Type A is passive (A←B). See Table 45 on page 160 for additional information.

Table 25. Summary of KTV tables

A→B or A←B	MAC generate/ verify	Data encrypt/ decrypt	PIN encrypt/ decrypt	Key wrap/unwrap
A→B (A active)	KTVM1 Table 46 on page 161	KTVC1 Table 48 on page 161	KTVP1 Table 50 on page 162	KTVW1 Table 52 on page 162
A←B (B active)	KTVM2 Table 47 on page 161	KTVC2 Table 49 on page 161	KTVP2 Table 51 on page 162	KTVW2 Table 53 on page 163

Table 26. KTV for MAC generate/verify, Type A active and Type B passive

Version (offset 0)	**Key type indicator (offset 2)**	**Algorithm indicator (offset 4)**	**Key length (offset 6)**	**Key usage indicator 1 (offset 8)**	**Key usage restriction 2 (offset 10)**	**RFU (offset 12)**	**Key direction variant indicator (offset 15)**
0	MAC	AES	AES-256	CMAC	'else'	-	A→B
00	00 00	00 02	01 00	00 01	00 00	00 00 00	01
KTVM1 = X'00 00 00 00 00 02 01 00 00 01 00 00 00 00 00 01'							

Table 27. KTV for MAC generate/verify, Type B active and Type A passive

Version (offset 0)	**Key type indicator (offset 2)**	**Algorithm indicator (offset 4)**	**Key length (offset 6)**	**Key usage indicator 1 (offset 8)**	**Key usage restriction 2 (offset 10)**	**RFU (offset 12)**	**Key direction variant indicator (offset 15)**
0	MAC	AES	AES-256	CMAC	'else'	-	A←B
00	00 00	00 02	01 00	00 01	00 00	00 00 00	10
KTVM2 = X'00 00 00 00 00 02 01 00 00 01 00 00 00 00 00 10'							

Table 28. KTV for data encryption (cipher), Type A active and Type B passive

Version (offset 0)	**Key type indicator (offset 2)**	**Algorithm indicator (offset 4)**	**Key length (offset 6)**	**Key usage indicator**	**Key usage restriction**	**RFU (offset 12)**	**Key direction variant**

				1 (offset 8)	2 (offset 10)		indicator (offset 15)
0	Cipher	AES	AES-256	CBC	'else'	-	A→B
00	00 01	00 02	01 00	00 02	00 00	00 00 00	01
KTVC1 = X'00 00 00 01 00 02 01 00 00 02 00 00 00 00 00 01'							

Table 29. KTV for data encryption (cipher), Type B active and Type A passive

Version (offset 0)	Key type indicator (offset 2)	Algorithm indicator (offset 4)	Key length (offset 6)	Key usage indicator 1 (offset 8)	Key usage restriction 2 (offset 10)	RFU (offset 12)	Key direction variant indicator (offset 15)
0	Cipher	AES	AES-256	CBC	'else'	-	A←B
00	00 01	00 02	01 00	00 02	00 00	00 00 00	10
KTVC2 = X'00 00 00 01 00 02 01 00 00 02 00 00 00 00 00 10'							

For PIN encryption key type, the key usage indicator 1 (offset 8) for ISO-4 format can have the value of ‘0000’ or ‘0002’ for a pair of KTVs. The caller is not allowed to mix pairs of KTVs because the KTV is used as the IV in the key creating process. This is the responsibility of the caller of the service.

KTV pair for PIN encryption key type with key usage indicator 1 with ‘0002’ value

Table 30. KTV for PIN encryption, Type A active and Type B passive

Version (offset 0)	Key type indicator (offset 2)	Algorithm indicator (offset 4)	Key length (offset 6)	Key usage indicator 1 (offset 8)	Key usage restriction 2 (offset 10)	RFU (offset 12)	Key direction variant indicator (offset 15)
0	PIN-Enc	AES	AES-256	ISO-4	'else'	-	A→B
00	00 03	00 02	01 00	00 02	00 00	00 00 00	01
KTVP1 = X'00 00 00 03 00 02 01 00 00 02 00 00 00 00 00 01'							

Table 31. KTV for PIN encryption, Type B active and Type A passive

Version (offset 0)	Key type indicator (offset 2)	Algorithm indicator (offset 4)	Key length (offset 6)	Key usage indicator 1 (offset 8)	Key usage restriction 2 (offset 10)	RFU (offset 12)	Key direction variant indicator (offset 15)
0	PIN-Enc	AES	AES-256	ISO-4	'else'	-	A←B
00	00 03	00 02	01 00	00 02	00 00	00 00 00	10
KTVP2 = X'00 00 00 03 00 02 01 00 00 02 00 00 00 00 00 10'							

KTV pair for PIN encryption key type with key usage indicator 1 with ‘0000’ value.

Table 32. KTV for PIN encryption, Type A active and Type B passive

Version (offset 0)	Key type indicator (offset 2)	Algorithm indicator (offset 4)	Key length (offset 6)	Key usage indicator	Key usage restriction	RFU (offset 12)	Key direction variant

				1 (offset 8)	2 (offset 10)		indicator (offset 15)
0	PIN-Enc	AES	AES-256	ISO-4	'else'	-	A→B
00	00 03	00 02	01 00	00 00	00 00	00 00 00	01
KTVP1 = X'00 00 00 03 00 02 01 00 00 00 00 00 00 00 00 01'							

Table 33. KTV for PIN encryption, Type B active and Type A passive

Version (offset 0)	Key type indicator (offset 2)	Algorithm indicator (offset 4)	Key length (offset 6)	Key usage indicator 1 (offset 8)	Key usage restriction 2 (offset 10)	RFU (offset 12)	Key direction variant indicator (offset 15)
0	PIN-Enc	AES	AES-256	ISO-4	'else'	-	A←B
00	00 03	00 02	01 00	00 00	00 00	00 00 00	10
KTVP2 = X'00 00 00 03 00 02 01 00 00 02 00 00 00 00 00 10'							

For Table 52 on page 162, entity Type A must use an AES EXPORTER key with usage of EXPTT31D, while entity Type B must use an IMPORTER key with usage of IMPTT31D. Key wrapping with key block protection (ISO TC 68/SC 2 Nxxxx, 2016-08-17, ISO DIS 20038):

Table 34. KTV for key wrapping, Type A active and Type B passive

Version (offset 0)	Key type indicator (offset 2)	Algorithm indicator (offset 4)	Key length (offset 6)	Key usage indicator 1 (offset 8)	Key usage restriction 2 (offset 10)	RFU (offset 12)	Key direction variant indicator (offset 15)
0	Key wrap	AES	AES-256	VARDRV-D	Maximum bit length of the protected keys	-	A→B
00	00 04	00 02	01 00	00 01	01 00	00 00 00	01
KTVW1 = X'00 00 00 04 00 02 01 00 00 01 01 00 00 00 00 01'							

For Table 53 on page 163, entity Type B must use an AES EXPORTER key with usage of EXPTT31D, while entity Type A must use an IMPORTER key with usage of IMPTT31D.

Table 35. KTV for key wrapping, Type B active and Type A passive

Version (offset 0)	Key type indicator (offset 2)	Algorithm indicator (offset 4)	Key length (offset 6)	Key usage indicator 1 (offset 8)	Key usage restriction 2 (offset 10)	RFU (offset 12)	Key direction variant indicator (offset 15)
0	Key wrap	AES	AES-256	VARDRV-D	Maximum bit length of the protected keys	-	A←B
00	00 04	00 02	01 00	00 01	01 00	00 00 00	10
KTVW2 = X'00 00 00 04 00 02 01 00 00 01 01 00 00 00 00 10'							

additional_derivation_data_length

Direction	Type
Input	Integer

Specifies the length in bytes of the additional_derivation_data parameter. The value must be between 0 and 2032 inclusive for CCA release 6.7, 7.4, and later. Otherwise, the value must be 0 and 24 inclusive.

The sum of the additional_derivation_data_length and the random_data_length cannot exceed 2048.

additional_derivation_data

Direction	Type
Input	String

Data to be used in the key generation or key derivation process.

The additional derivation data concatenated with the random data cannot exceed 2048 for CCA releases 6.7, 7.4, and later. Otherwise the random data cannot exceed 40 bytes.

random_data_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the random_data parameter. The value must be between 16 and 40 inclusive. The sum of the additional_derivation_data_length and the random_data_length cannot exceed 2048 for CCA releases 6.7, 7.4, and later. Otherwise, the random data cannot exceed 40 bytes.

When keyword GENERATE is specified in the rule array, this is an input and an output parameter. On input, this value specifies the number of bytes of data to use as the random data portion of the diversification data used in diversifying the first key of a key pair. On output, the returned value indicates the number of bytes of data actually returned in the random_data variable.

When keyword DERIVE is specified in the rule array, this is an input only parameter. On input, this value specifies the number of bytes of data to use as the random data portion of the diversification data used in diversifying the second key of a key pair. To produce the desired results, this value must be the same length returned by a previous associated GENERATE function call.

random_data

Direction	Type
Input/Output	String

The random data used in the diversification process.

When the GENERATE function is specified, on input, this variable is ignored, and on output, this variable contains the random data created and used to diversify the first output key of a key pair.

When the DERIVE function is specified, on input, this variable must contain the random data previously created during a previous GENERATE function and is used to diversify the second output key of a key pair.

Note: For a given pair of output keys, the DERIVE function must provide the same random data and additional_derivation_data value as the GENERATE function used.

The additional derivation data concatenated with the random data cannot exceed 2048 for CCA releases 6.7, 7.4, and later. Otherwise, the random data cannot exceed 40 bytes.

output_key_identifier_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the buffer for the output_key_identifier parameter. On input, the value is the size of the buffer. The maximum length is 725. On output, the value is the length of the key token returned in the output_key_identifier parameter.

output_key_identifier

Direction	Type
Output	String

The buffer to receive the generated key. The key attributes are identified by the key_type_vector parameter.

When the kdk_key_identifier is compliant-tagged, a compliant-tagged key token will be created.

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS.

Access control points

The Diversify Directed Key access control point in the domain role controls the function of this service.

The access controls for rule array keywords are listed in the table:

Diversification process rulearray keyword	Function rule-array keyword	Access control
KDFFM	DERIVE	Diversify Directed Key – allow KDFFM DERIVE
	GENERATE	Diversify Directed Key – allow KDFFM GENERATE

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 36. Diversify Directed Key required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC		This service is not supported.
IBM System z10 EC IBM System z10 BC		This service is not supported.
IBM zEnterprise 196 IBM zEnterprise 114		This service is not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12		This service is not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	This service requires the July 2019 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. The additional_derivation_data_length must not exceed 24.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	This service requires the July 2019 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. The additional_derivation_data_length must not exceed 24.
	Crypto Express6 CCA Coprocessor	This service requires the December 2018 or later licensed internal code (LIC). Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC). The additional_derivation_data_length parameter support requires the require CCA release 6.7 or later licensed internal code.
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. The additional_derivation_data_length must not exceed 24.
	Crypto Express6 CCA Coprocessor	The additional_derivation_data_length parameter support requires the require CCA release 6.7 or later licensed internal code.
	Crypto Express7 CCA Coprocessor	The additional_derivation_data_length parameter support requires the CCA release 7.4 or later licensed internal code.

Random Number Generate (CSNBRNG, CSNERNG, CSNBRNGL and CSNERNGL)

The callable service uses a cryptographic feature to generate a random number. The foundation for the random number generator is a time variant input with a very low probability of recycling.

There are two forms of the Random Number Generate callable service. One version returns an 8-byte random number. The second version allows the caller to specify the length of the random number.

The callable service names for AMODE(64) invocation are CSNERNG and CSNERNGL.

Format

```
CALL CSNBRNG(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              form,
              random_number )

CALL CSNBRNGL(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              key_identifier_length,
              key_identifier,
              random_number_length,
              random_number )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

form

Direction	Type
Input	Character String

The 8-byte keyword for the CSNBRNG service that defines the characteristics of the random number should be left-justify and pad on the right with blanks. The keywords are listed in Table 140 on page 350.

Table 37. Keywords for the Form Parameter

Keyword	Meaning
EVEN	Generate a 64-bit random number with even parity in each byte.
ODD	Generate a 64-bit random number with odd parity in each byte.
RANDOM	Generate a 64-bit random number.

Parity is calculated on the 7 high-order bits in each byte and is presented in the low-order bit in the byte.

rule_array_count

Direction	Type
Input	Integer

The number of keywords for the CSNBRNGL service you are supplying in the rule_array parameter. The value must be 1 or 2.

rule_array

Direction	Type
Input	String

The keyword for the CSNBRNGL service that provides control information to the callable service. The recovery method is the method to use to recover the symmetric key. The keyword is left-justified in an 8-byte field and padded on the right with blanks. All keywords must be in contiguous storage.

Table 38. Keywords for Random Number Generate Control Information

Keyword	Meaning
Requested service (one, required)	
EVEN	Specifies that each generated random byte is adjusted for even parity.
ODD	Specifies that each generated random byte is adjusted for odd parity.
RANDOM	Specifies that each generated random byte is not adjusted for parity.
RT-KRD	Specifies that the generated random number is returned formatted as a TR-34 Key Receiving Device Random Number Token (RT-KRD). The token requires 21 additional bytes for

	encoding and overhead. The random number is not adjusted for parity. Note the maximum size of the token that is usable with the service that the token is planned to be used with. If the maximum size is 200 bytes, the maximum random number size is 179 bytes.
Encryption Process (one, optional) Not valid with the RT-KRD keyword	
TDES-CBC	Specifies to return the random number encrypted using the DES key specified in the key_identifier parameter. Note: A CCA Crypto Express coprocessor must be active to get encrypted output.

key_identifier_length

Direction	Type
Input	Integer

The length of the key_identifier parameter in bytes. When the rule array keyword TDES-CBC is specified, the value must be 64.

Otherwise, the value must be 0.

key_identifier

Direction	Type
Input/Output	String

The identifier of the key to encrypt the random number. The key identifier is an operational token or the key label of an operational token in key storage. When the TDES-CBC keyword is specified, the key algorithm of this key must be DES, the key type must be CIPHER or ENCIPHER, and the key must be a double-length or triple-length key.

When the key_identifier_length parameter is 0, this parameter is ignored.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

random_number_length

Direction	Type
Input/Output	Integer

This parameter contains the desired length of the random_number that is returned by the CSNBRNGL callable service. The minimum value is 1 byte; the maximum value is 8192 bytes.

When the requested service keyword is TDES-CBC, the value must be a multiple of 8. The maximum value is 1024.

When the requested service keyword is RT-KRD:

- On input, this value is the number of bytes of the random number requested plus 21 bytes for the DER encoding of the token. Note the maximum size of the token that is

usable with the service that the token is planned to be used with. If the maximum size is 200 bytes, the maximum random number size is 179 bytes.
- On output, the value will be the actual size of the token returned.

random_number

Direction	Type
Output	String

The generated number returned by the CSNBRNG callable service is stored in an 8-byte variable.

The generated number returned by the CSNBRNGL callable service is stored in a variable that is at least random_number_length bytes long.

When the requested service keyword is RT-KRD, the TR-34 Key Receiving Device Random Number Token is returned.

Usage notes

If the CSF.CSFSERV.AUTH.CSFRNG.DISABLE SAF resource profile is defined in the XFACILIT SAF resource class, no SAF authorization checks will be performed against the CSFSERV class when using this service. If CSF.CSFSERV.AUTH.CSFRNG.DISABLE is not defined, the SAF authorization check will be performed. Disabling the SAF check may improve the performance of your application.

Access control points

The CSNBRNG service requires that the **Key Generate – SINGLE-R** access control point is enabled. The CSNBRNGL service is not controlled by any access control.

The use of the TDES-CBC rule array keyword requires the **Random Number Generate Long – TDES-CBC** access control be enabled.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 39. Random Number Generate required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.
IBM System z10 EC IBM System z10 BC	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.
IBM zEnterprise 196 IBM zEnterprise 114	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.
IBM z13 IBM z13s	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.
IBM z14 IBM z14 ZR1	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.

	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.
IBM z15 IBM z15 T02	CP Assist for Cryptographic Functions	Rule array keywords TDES-CBC is not supported.
	Crypto Express7 CCA Coprocessor	Rule array keyword TDES-CBC requires the CCA release 7.4 or later licensed internal code (LIC).

Symmetric Key Export (CSNDSYX and CSNFSYX)

Use the Symmetric Key Export callable service to transfer an operational AES, DES, or HMAC key in a CCA key token from encryption under a master key to encryption under an RSA public key or AES EXPORTER key.

For an RSA-enciphered output key, the key is returned as an opaque data buffer or in an external variable-length symmetric key-token. If the key is returned as an opaque data buffer, the Symmetric Key Import service can be used along with the associated RSA private-key to import the key back into an operational symmetric key-token. If the key is returned in an external variable-length symmetric key-token, the Symmetric Key Import2 service can be used along with the associated RSA private-key to import the key.

For an AES-enciphered output key, the key is returned in an external variable-length symmetric key- token. The Symmetric Key Import2 service can be used along with its associated AES IMPORTER key- encrypting key to import the key. The usage attributes of the IMPORTER key must allow IMPORT.

Table 40 and 41 show which formatting methods can be used for each type of key token and a description of the enciphered key returned.

Table 40. CSNDSYX key formatting for fixed length AES and DES key tokens

Operational source key-token	**Key-formatting method keyword**			
	AESKWCV	**PKCS-1.2**	**PKCSOAEP**	**ZERO-PAD**
AES DATA	Not supported.	The output key is returned as an opaque data buffer after being formatted using the RSAES-PKCS1-v1_5 encryption / decryption scheme of the RSA PKCS #1 v2.0 standard and enciphered using the RSA public-key provided as a transport key.	The output key is returned as an opaque data buffer after being formatted using the RSAES-OAEP encryption / decryption scheme of the RSA PKCS #1 v2.0 standard and enciphered using the RSA public-key provided as a transport key.	The output key is returned as an opaque data buffer after the key is right-aligned, padded on the left to the necessary block length with bits valued to zero, and enciphered using the RSA public-key provided as a transport key.
DES DATA	The output key is returned in an external variable-length DES key-token with control vector after being enciphered using the AES EXPORTER key provided as the transport key.			
DES key types other than DATA	The output key is returned in an external variable-length DES key-token with control vector after being enciphered using the AES EXPORTER key provided as the	Not supported.	Not supported.	Not supported.

	transport key.			

Table 41. CSNDSYX key formatting for variable length AES and HMAC key tokens

Operational source key-token	**Key-formatting method keyword**		
	AESWK	**PKOAEP2**	**CKM-RAKW**
AES	The output key is returned in an external variable-length AES key-token after being enciphered using he AES EXPORTER key provided as the transport key.	The output key is returned in an external variable length AES key token after being formatted using the RSAES-OAEP encryption / decryption scheme of the RSA PKCS #1 v2.1 standard and enciphered using the RSA public-key provided as a transport key.	The output key is returned as output structure corresponding to the output from the PKCS#11 mechanism CKM_RSA_AES_KEY_WRAP and enciphered using the RSA public-key provided as a transport key.
HMAC	Same as variable-length AES source key-token, except that the output key is returned in an external variable-length HMAC key-token.	Same as variable-length AES source key-token, except that the output key is returned in an external variable-length HMAC key-token.	Not supported.

Notes:
1. For keywords PKCS-1.2, PKCSOAEP, and PKOAEP2, see “Formatting hashes and keys in public-key cryptography” on page xxx.
2. The RSA PKCS #1 v2.0 standard for the RSAES-PKCS1-v1_5 encryption/decryption scheme is formerly known as block-type 02 format.
3. PKCSOAEP and PKOAEP2 are the only key formatting methods that use a hash method. PKCSOAEP and PKOAEP2 can specify either SHA-1 or SHA-256. PKOAEP2 can also specify SHA-384 or SHA-512.

The callable service name for AMODE(64) is CSNFSYX.

Format

```
CALL CSNDSYX(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              source_key_identifier_length,
              source_key_identifier,
              transporter_key_identifier_length,
              transporter_key_identifier,
              enciphered_key_length,
              enciphered_key)
```

Parameters

return_code

Direction	**Type**
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you are supplying in the rule_array parameter. Value may be 1, 2, or 3.

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service. Table 42 on page 376 lists the keywords. Each keyword is left-justified in 8-byte fields and padded on the right with blanks. All keywords must be in contiguous storage.

Table 42 Keywords for Symmetric Key Export Control Information

Keyword	Meaning
Token Algorithm (One keyword, optional)	
AES	The key being exported is an AES key. If source_key_identifier is a variable-length symmetric key token or label, only the PKOAEP2 and AESKW key formatting methods are supported.
DES	The key being exported is a DES key. This is the default.
HMAC	The key being exported is an HMAC key. Only the PKOAEP2 and AESKW key formatting methods are supported.
Key Formatting method (One required)	

AESKW	Specifies that the key is to be formatted using AESKW and placed in an external variable length CCA token. The transport_key_identifier must be an AES EXPORTER. This rule is not valid with the DES Algorithm keyword or with AES DATA (version X'04') keys.
AESKWCV	Specifies that the key is to be formatted using AESKW and placed in a symmetric variable length CCA token of type DESUSECV. The transport_key_identifier must be an AES EXPORTER key. The DES control vector and other significant token information will be in the associated data section of the variable length key token. Only valid with the DES token algorithm.
CKM-RAKW	Specifies to return the key in an external AES wrapped PKCS#11 object. The variable-length symmetric key-token will be returned in an output structure corresponding to the output from PKCS#11 mechanism CKM_RSA_AES_KEY_WRAP. Valid only with the AES algorithm.
PKCSOAEP	Specifies to format the key according to the method in RSA DSI PKCS #1V2 OAEP. The default hash method is SHA-1. Use the SHA-256 keyword for the SHA-256 hash method.
PKCS–1.2	Specifies to format the key according to the method found in RSA DSI PKCS #1 block type 02 to recover the symmetric key.
PKOAEP2	Specifies to format the key according to the method found in RSA DSI PKCS #1 v2.1 RSAES-OAEP documentation. Not valid with DES algorithm or with AES DATA (version X'04') keys. A hash method is required.
ZERO-PAD	The clear key is right-justified in the field provided, and the field is padded to the left with zeros up to the size of the RSA encryption block (which is the modulus length).
Hash Method (One, optional for PKCSOAEP, required for PKOAEP2. Not valid with any other Key Formatting method)	
SHA-1	Specifies to use the SHA-1 hash method to calculate the OAEP message hash. This is the default for PKCSOAEP.
SHA-256	Specifies to use the SHA-256 hash method to calculate the OAEP message hash.
SHA-384	Specifies to use the SHA-384 hash method to calculate the OAEP message hash. Not valid with PKCSOAEP.
SHA-512	Specifies to use the SHA-512 hash method to calculate the OAEP message hash. Not valid with PKCSOAEP.
Certificate validation method (One required when the input is an X.509 certificate. Otherwise, must not be specified.)	
RFC-2459	Validate the certificate using the semantics of RFC-2459.
RFC-3280	Validate the certificate using the semantics of RFC-3280.
RFC-5280	Validate the certificate using the semantics of RFC-5280.
RFC-ANY	Attempt to validate the certificate by first using the semantics of RFC-2459, then the semantics of RFC-3280, and finally, the semantics of RFC-5280.
Public Key Infrastructure Usage (One optional when the input is an X.509 certificate. Otherwise, must not be specified.)	
PKI-CHK	Specifies that the X.509 certificate is to be validated against the trust chain of the PKI hosted in the adapter. This requires that the CA credentials have been installed into all coprocessors using the Trusted Key Entry workstation. This is the default.
PKI-NONE	Specifies that the X.509 certificate is not to be validated against the trust chain of the PKI hosted in the adapter. This is suitable if the certificate has been validated using host-based PKI services or if using a self-signed certificate.

source_key_identifier_length

Direction	Type
Input	Integer

The length in bytes of the source_key_identifier parameter. The minimum size is 64 bytes. If the source_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

source_key_identifier

Direction	Type
Input/Output	String

The label or internal token of a secure AES DATA (version X‘04’), DES DATA, or variable-length symmetric key token to encrypt under the supplied RSA public key or a secure AES or DES key token to encrypt under the supplied AES EXPORTER key. The key in the key identifier must match the algorithm in the rule_array. DES is the default algorithm.

transporter_key_identifier_length

Direction	Type
Input	Integer

The key to be exported and wrapped by the transport_key_identifier. The key identifier is an operational token or the key label of an operational token in key storage.

The key in the key identifier must match the algorithm in the rule_array. DES is the default algorithm.

For formatting method rules PKCSOAEP, PKCS-1.2, and ZERO-PAD, the source key is an AES or DES DATA key in a fixed-length key token.

For rule AESKWCV, the source key is a DES key of any type in a fixed-length key token.

For rules AESKW and PKOAEP2, the source key is an AES or HMAC key of any type in a variable-length key token.

For rule CKM-RAKW, the source key is an AES CIPHER key in a variable-length key token.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

transporter_key_identifier

Direction	Type
Input	String

The key to wrap the source key in a formatted data buffer or external key token. The key identifier is an operational token, the key label of an operational token in key storage, or an X.509 certificate containing the public key.

When the AESKW or AESKWCV key formatting method is specified, this parameter must be an AES EXPORTER key token or label with the EXPORT bit on in the key-usage field.

The key usage wrap algorithm control must match the algorithm of the source key. The key usage wrap class control must match the class of the source key.

Otherwise, this parameter must be the token or label of an RSA public or private key token, or the X.509 certificate containing the RSA public key.

Certificates may be PEM-formatted EBCDIC text or DER-encoded. The certificate may either have no key usage attribute, or it must have the following usage: keyEncipherment.

When the identifier is an AES EXPORTER and the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

Certificates may be PEM-formatted EBCDIC text or DER-encoded. The certificate may either have no key usage attribute, or it must have the following usage: keyEncipherment.

enciphered_key_length

Direction	Type
Input/Output	Integer

The length of the enciphered_key parameter. This is updated with the actual length of the enciphered_key generated. The maximum size you can specify in this parameter is 900 bytes, although the actual key length may be further restricted by your hardware configuration (as shown in Table 159 on page 380).

enciphered_key

Direction	Type
Output	String

The exported key in the specified format wrapped by the RSA public or AES EXPORTER key specified in the transporter_key_identifier field.

Usage notes

If ICSF is configured to audit the lifecycle of tokens (for example, AUDITKEYLIFECKDS(TOKEN(YES),...) is specified) and a token is passed as input to be exported, a request is made to the Crypto Express Coprocessor to generate the key fingerprint to be used for auditing the exported key.

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

If an RSA public key is specified as the transporter_key_identifier, the hardware configuration sets the limit on the modulus size of keys for key management; thus, this service will fail if the RSA key modulus bit length exceeds this limit.

When wrapping an AES key with an RSA public key, the RSA key used must have a modulus size greater than or equal to the total PKOAEP2 message bit length (key size + total overhead).

Table 43. Minimum RSA modulus strength required to contain a PKOAEP2 block when exporting an AES key

AES key size	Total message sizes (and therefore minimum RSA key size) when the Hash Method is:			
	SHA-1	SHA-256	SHA-384	SHA-512
128 bits	736 bits	928 bits	1184 bits	1440 bits
192 bits	800 bits	992 bits	1248 bits	1504 bits

256 bits	800 bits	1056 bits	1312 bits	1568 bits

Access control points

The following table shows the access control points in the domain role that control the function of this service.

Table 44. Required access control points for Symmetric Key Export

Key formatting method	Token Algorithm	Access control point
PKCSOAEP	AES	Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2
	DES	Symmetric Key Export - DES, PKCS-1.2
PKCS-1.2	AES	Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2
	DES	Symmetric Key Export - DES, PKCS-1.2
ZERO-PAD	AES	Symmetric Key Export - AES, ZEROPAD
	DES	Symmetric Key Export - DES, ZEROPAD
PKOAEP2	HMAC	Symmetric Key Export - HMAC, PKOAEP2
	AES	Symmetric Key Export - AES, PKOAEP2
AESKW	AES or HMAC	Symmetric Key Export - AESKW
AESKWCV	DES	Symmetric Key Export – AESKWCV
CKM-RAKW	AES	Symmetric Key Export - CKM-RAKW

If the transport key identifier is a weaker key than the key being exported, then:

- the service will fail if the **Prohibit weak wrapping - Transport keys** access control point is enabled.
- the service will complete successfully with a warning return code if the **Warn when weak wrap - Transport keys** access control point is enabled.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 45. Symmetric Key Export required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the November 2007 or later licensed internal code (LIC). Encrypted AES key support requires the November 2008 or later licensed internal code (LIC). The AESKW, AESKWCV, CKM-RAKW, HMAC, and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for

		PKCSOAEP. Triple-length DES keys are not supported. Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048- bit to 4096-bit requires the November 2007 or later licensed internal code (LIC). Encrypted AES key support requires the November 2008 or later licensed internal code (LIC). The AESKW, AESKWCV, CKM-RAKW, HMAC, and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP. Triple-length DES keys are not supported. Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express3 Coprocessor	The AESKW, AESKWCV, CKM-RAKW, HMAC, and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP. Triple-length DES keys are not supported. Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.

IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code. Variable-length AES Keys, the AESKW method, and PKCSOAEP with the SHA-256 hash method require the September 2011 or later licensed internal code (LIC). HMAC key support requires the November 2010 or later licensed internal code (LIC). The AESKWCV and CKM-RAKW keywords are not supported. Triple-length DES keys are not supported. Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code. AESKWCV requires the September 2013 or later licensed internal code (LIC). Triple-length DES keys are not supported. Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported. Rule array keyword CKM-RAKW is not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Triple-length DES keys require the July 2019 or later licensed internal code (LIC). Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.

		Rule array keyword CKM-RAKW is not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Triple-length DES keys require the December 2018 or later licensed internal code (LIC). Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported. Rule array keyword CKM-RAKW is not supported.
	Crypto Express6 CCA Coprocessor	Triple-length DES keys require the December 2018 or later licensed internal code (LIC). Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE require the July 2019 or later licensed internal code (LIC). X.509 certificates require the July 2019 or later licensed internal code (LIC). Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC). Rule array keyword CKM-RAKW is not supported.
IBM z15 IBM z15 T02	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. X.509 certificates are not supported. Rule array keyword CKM-RAKW is not supported.
	Crypto Express6 CCA Coprocessor	Rule array keyword CKM-RAKW is not supported.
	Crypto Express7 CCA Coprocessor	Rule array keyword CKM-RAKW requires the CCA release 7.4 or later licensed internal code (LIC).

Symmetric Algorithm Decipher (CSNBSAD or CSNBSAD1 and CSNESAD or CSNESAD1)

The symmetric algorithm decipher callable service deciphers data with the AES algorithm. Encryption modes supported are Cipher Block Chaining (CBC) mode, Electronic Code Book (ECB) mode, and Galois/ Counter Mode (GCM).

You can specify that the clear text data was padded before encryption using the method described in the PKCS standards. In this case, the callable service will remove the padding bytes and return the unpadded clear text data. PKCS padding is described in “PKCS padding method” on page 1468.

The callable service names for AMODE(64) invocation are CSNESAD and CSNESAD1.

Choosing between CSNBSAD and CSNBSAD1 or CSNESAD and CSNESAD1

CSNBSAD, CSNBSAD1, CSNESAD, and CSNESAD1 provide identical functions. When choosing which service to use, consider this:

- CSNBSAD and CSNESAD require the cipher text and plaintext to reside in the caller’s primary address space. Also, a program using CSNBSAD adheres to the IBM Common Cryptographic Architecture: Cryptographic Application Programming Interface.
- CSNBSAD1 and CSNESAD1 allow the cipher text and plaintext to reside either in the caller’s primary address space or in a data space. This can allow you to decipher more data with one call. However, a program using CSNBSAD1 and CSNESAD1 does not adhere to the IBM CCA: Cryptographic API and may need to be modified prior to it running with other cryptographic products that follow this programming interface.

For CSNBSAD1 and CSNESAD1, cipher_text_id and clear_text_id are access list entry token (ALET) parameters of the data spaces containing the cipher text and plaintext.

Format

```
CALL CSNBSAD(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             key_parms_length,
             key_parms,
             block_size,
             initialization_vector_length,
             initialization_vector,
             chain_data_length,
             chain_data,
             cipher_text_length,
             cipher_text,
             clear_text_length,
             clear_text,
             optional_data_length,
             optional_data)
```

```
CALL CSNBSAD1(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              key_length,
              key_identifier,
              key_parms_length,
              key_parms,
              block_size,
              initialization_vector_length,
              initialization_vector,
              chain_data_length,
              chain_data,
              cipher_text_length,
              cipher_text,
              clear_text_length,
              clear_text,
              optional_data_length,
              optional_data,
              cipher_text_id,
              clear_text_id)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Ignored	Integer

This field is ignored. It is recommended to specify 0 for this parameter.

exit_data

Direction	Type

Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value may be 2, 3 or 4.

rule_array

Direction	Type
Input	String

An array of 8-byte keywords providing the processing control information. The keywords must be in contiguous storage, left-justified and padded on the right with blanks.

Table 46. Symmetric Algorithm Decipher Rule Array Keywords

Keyword	Meaning
Algorithm (required, one keyword)	
AES	Specifies that the Advanced Encryption Standard (AES) algorithm is to be used. The block size is 16 bytes. The key length may be 16, 24, or 32 bytes.
Processing Rule (optional, one keyword)	
CBC	Performs encryption in cipher block chaining (CBC) mode. The text length must be a multiple of the AES block size (16-bytes). This is the default value.
ECB	Performs encryption in electronic code book (ECB) mode. The text length must be a multiple of the AES block size (16-bytes).
GCM	Performs Galois/Counter mode decryption. The plaintext will have the same length as the ciphertext. Additionally, the authentication tag will be verified before the data is returned.
PKCS-PAD	Performs encryption in cipher block chaining (CBC) mode. The ciphertext length must be an exact multiple of 16 bytes. Padding is removed from the plaintext and the text length is reduced to the original value. This rule should be specified only when there is one request or on the last request of a sequence of chained requests.
X9.23PAD	Specifies that the cleartext was padded according to the PKCS padding scheme. Performs encryption in cipher block chaining (CBC) mode. The ciphertext length must be an exact multiple of 16 bytes. Padding is removed from the plaintext and the text length is reduced to the original value. This rule should be specified only when there is one request or on the last request of a sequence of chained requests.
Key Rule (required, one keyword)	
KEYIDENT	This indicates that the value in the key_identifier parameter is either an internal key token or the label of a key token in the CKDS. The key must be a secure AES key, that is, enciphered under the current master key.
ICV Selection (optional for CBC and PKCS-PAD, required for GCM, one keyword)	
INITIAL	This specifies that this is the first request of a sequence of chained requests and indicates that the initialization vector should be taken

	from the initialization_vector parameter. This is the default value for CBC and PKDS-PAD. This keyword is not valid with processing rule GCM.
CONTINUE	This specifies that this request is part of a sequence of chained requests, and is not the first request in that sequence. The initialization vector will be taken from the work area identified in the chain_data parameter. This keyword is only valid for processing rules CBC or PKCS-PAD.
ONLY	Specifies that this is the only request and indicates that the initialization vector should be taken from the initialization_vector parameter. Only valid with the processing rule GCM.

key_identifier_length

Direction	Type
Input	Integer

The length of the key_identifier parameter in bytes. The length must be 64 bytes for a fixed-length (version X’04’) token or a CKDS label, or between the actual length of the token and 725 for a variable-length (version X’05’) token.

key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the text. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be AES, the key type must be DATA (fixed-length token, version X'04') or CIPHER (variable-length token, version X'05'). For the CIPHER key, the key usage must indicate DECRYPT and the appropriate mode of encryption (CBC, ECB, GCM, or ANY-MODE).

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

key_parms_length

Direction	Type
Input	Integer

The length of the key_parms parameter in bytes.

For the GCM processing rule, this is the length of the authentication tag to be verified. Valid lengths are 4, 8, 12, 13, 14, 15, and 16, but using a length of 4 or 8 is strongly discouraged.

For all other processing rules, the value must be zero.

key_parms

Direction	Type
Input	String

The key_parms parameter contains key related parameters.

For the GCM processing rule, key_parms will contain an authentication tag to be verified for the provided ciphertext (cipher_text parameter) and additional authenticated data (optional_data parameter). You must specify the same key_parms generated when the text was enciphered.

Otherwise, this parameter is ignored.

block_size

Direction	Type
Input	Integer

The block size for the cryptographic algorithm. AES requires the block size to be 16.

initialization_vector_length

Direction	Type
Input	Integer

The length of the initialization_vector parameter in bytes. For CBC, PKCS-PAD, and X9.23PAD, the length must be equal to the block length for the algorithm specified, 16. For the GCM processing rule, NIST recommends a length of 12, but tolerates any non-zero length up to a maximum of 232-1.

This parameter is ignored when the process rule is ECB.

initialization_vector

Direction	Type
Input/Output	String

This parameter contains the initialization vector (IV) for CBC mode decryption. This includes CBC, GCM, PKCS-PAD, and X9.23PAD processing rule keywords. The IV must be the same value used when the data was encrypted.

This parameter is ignored when the process rule is ECB.

chain_data_length

Direction	Type
Input	Integer

The length of the chain_data parameter in bytes. On input, it contains the length of the buffer provided with parameter chain_data. On output, it is updated with the length of the data returned in the chain_data parameter.

For CBC, the value must be at least 32. For ECB and GCM, the parameter is ignored.

chain_data

Direction	Type
Input/Output	String

A buffer that is used as a work area for sequences of chained symmetric algorithm decipher requests. The exact content and layout of chain_data is not described. Your application program must not change the data in this string.

When the keyword INITIAL is used, this is an output parameter and receives data that is needed when deciphering the next part of the input data. When the keyword CONTINUE is used, this is an input/output parameter; the value received as output from the previous call in the sequence is provided as input to this call, and in turn, this call will return new chain_data that will be used as input on the next call. When CONTINUE is used, both the data (chain_data parameter) and the length (chain_data_length parameter) must be the same values that were received in these parameters as output on the preceding call to the service in the chained sequence.

For ECB and GCM, this parameter is ignored.

cipher_text_length

Direction	Type
Input	Integer

The length of the cipher text. For processing rules CBC, ECB, PKCS-PAD, and X9.23PAD, the length must be a multiple of the algorithm block size. The maximum length is 2^{32}-1.

For GCM, the value may be zero.

When the Crypto Express adapter is a CEX5 or CEX6, the maximum value is 2^{29}-1. When the Crypto Express adapter is a CEX7, the maximum value is 2^{32}-1.

cipher_text

Direction	Type
Input/Output	String

The text to be deciphered.

clear_text_length

Direction	Type
Input/Output	Integer

On input, this parameter specifies the size of the storage pointed to by the clear_text parameter. On output, this parameter has the actual length of the text stored in the clear_text parameter.

If process rule PKCS-PAD or X9.23PAD is used, the clear text length will be less than the cipher text length since padding bytes are removed.

clear_text

Direction	Type
Output	String

The deciphered text the service returns.

optional_data_length

Direction	Type
Input	Integer

The length of the optional_data parameter in bytes. For the GCM processing rule, this parameter contains the length of the Additional Authenticated Data (AAD). The value may be 0 to 2^{32}-1.

For all other processing rules, the value must be 0.

optional_data

Direction	Type
Input	String

Optional data required by a specified algorithm or processing mode. For the GCM processing rule, this parameter contains the Additional Authenticated Data (AAD). For all other processing rules, this field is ignored.

You must specify the same optional_data used when the text was enciphered.

cipher_text_id

Direction	Type
Input	Integer

For CSNBSAD1 and CSNESAD1 only, the ALET of the dataspace in which the cipher_text parameter resides.

clear_text_id

Direction	Type
Input	Integer

For CSNBSAD1 and CSNESAD1 only, the ALET of the dataspace in which the clear_text parameter resides.

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

The clear_text and cipher_text parameters may be in any dataspace. The initialization_vector and optional_data parameters must be in the caller's address space (primary).

Access control point

The **Symmetric Algorithm Decipher - secure AES keys** access control point controls the function of this service. Use of the GCM processing rule requires that the **Symmetric Algorithm Decipher – Galois/Counter mode AES** access control is enabled.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 47. Symmetric Algorithm Decipher required hardware

Server	Required cryptographic hardware	Restrictions

IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	Secure AES key support requires the Nov. 2008 or later licensed internal code (LIC). Keywords GCM, ONLY, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor Crypto Express3 Coprocessor	Secure AES key support requires the Nov. 2008 or later licensed internal code (LIC). Keywords GCM, ONLY, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	AES Variable-length Symmetric Internal Key Tokens require the Sep. 2011 or later licensed internal code (LIC). Keywords GCM, ONLY, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Keywords GCM, ONLY, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. Rule array keyword X9.23PAD is not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. Rule array keyword X9.23PAD is not supported.
	Crypto Express6 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Rule array keyword X9.23PAD requires the CCA release 6.7 or later licensed internal code (LIC).
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. Rule array keyword X9.23PAD is not supported.
	Crypto Express6 CCA Coprocessor	Rule array keyword X9.23PAD requires the CCA release 6.7 or later licensed internal code (LIC).
	Crypto Express7 CCA Coprocessor	Rule array keyword X9.23PAD requires the CCA release 7.4 or later licensed internal code (LIC).

Symmetric Algorithm Encipher (CSNBSAE or CSNBSAE1 and CSNESAE or CSNESAE1)

The symmetric algorithm encipher callable service enciphers data with the AES algorithm. Encryption modes supported are Cipher Block Chaining (CBC) mode, Electronic Code Book (ECB) mode, and Galois/Counter Mode (GCM).

The symmetric algorithm encipher service supports the Australian Payment Network (APN) standards to generate and verify MACs and related processing as defined in AS2805.5.4.

- To generate a MAC – Processing rule A28MACGN

  Parameters:
  key_identifier: double-length DES MAC key
  key_parms: double-length DES CIPHE R key
  clear_text: clear message text
  chain_data:
  Input: starting MAC residue, encrypted by the key in the key_parms parameter
  Output: Final MAC residue, encrypted by the key in the key_parms parameter
  cipher_text:
  Output: 8-byte MAC

- To verify a MAC – Processing rule A28MACVR

  Parameters:
  key_identifier: double-length DES MAC key
  key_parms: double-length DES CIPHE R key
  clear_text: clear message text
  chain_data:
  Input: starting MAC residue as indicated by the Residue value keyword, encrypted by the key in the key_parms parameter
  Output: Final MAC residue, encrypted by the key in the key_parms parameter
  cipher_text:
  Input: 8-byte MAC to verify

- One Way Function processing
  - Processing rule A28OWFEC

    Parameters:
    key_identifier: double-length DES EXPORTER key
    key_parms: double-length DES CIPHE R key
    chain_data: input value ECB wrapped by the key specified in the key_parms field
    cipher_text: output of the OWF

  - Processing rule A28OWFCL

    Parameters:
    key_identifier: double-length DES CIPHER key
    chain_data: clear input value
    cipher_text: output of the OWF

The callable service names for AMODE(64) invocation are CSNESAE and CSNESAE1

Choosing between CSNBSAE and CSNBSAE1 or CSNESAE and CSNESAE1

CSNBSAE, CSNBSAE1, CSNESAE, and CSNESAE1 provide identical functions. When choosing which service to use, consider this:

- CSNBSAE and CSNESAE require the cipher text and plaintext to reside in the caller's primary address space. Also, a program using CSNBSAE adheres to the IBM Common Cryptographic Architecture: Cryptographic Application Programming Interface.
- CSNBSAE1 and CSNESAE1 allow the cipher text and plaintext to reside either in the caller's primary address space or in a data space. This can allow you to encipher more data with one call. However, a program using CSNBSAE1 and CSNESAE1 does not adhere to the IBM CCA: Cryptographic API and may need to be modified prior to it running with other cryptographic products that follow this programming interface.

For CSNBSAE1 and CSNESAE1, cipher_text_id and clear_text_id are access list entry token (ALET) parameters of the data spaces containing the cipher text and plaintext.

Format

```
CALL CSNBSAE(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             key_parms_length,
             key_parms,
             block_size,
             initialization_vector_length,
             initialization_vector,
             chain_data_length,
             chain_data,
             clear_text_length,
             clear_text,
             cipher_text_length,
             cipher_text,
             optional_data_length,
             optional_data)

CALL CSNBSAE1(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             key_parms_length,
             key_parms,
             block_size,
             initialization_vector_length,
             initialization_vector,
```

```
chain_data_length,
chain_data,
clear_text_length,
clear_text,
cipher_text_length,
cipher_text,
optional_data_length,
optional_data,
clear_text_id,
cipher_text_id)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Ignored	Integer

This field is ignored. It is recommended to specify 0 for this parameter.

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value may be 2, 3, 4, or 5.

rule_array

Direction	Type
Input	String

This keyword provides control information to the callable service. The keywords must be eight bytes of contiguous storage with the keyword left-justified in its 8-byte location and padded on the right with blanks.

Table 48. Symmetric Algorithm Encipher Rule Array Keywords

Keyword	Meaning
Algorithm (required, one keyword)	
AES	Specifies that the Advanced Encryption Standard (AES) algorithm will be used. The block size is 16-bytes, and the key length may be 16-, 24-, or 32-bytes (128-, 192-, 256-bits).
DES	Specifies use of the Data Encryption Standard (DES) as the encryption algorithm. Only valid with processing rule A28MACGN, A28MACVR, A28OWFEC, and A28OWFCL keywords.
Processing Rule (one, required when the DES algorithm keyword is specified, optional when the AES keyword is specified)	
Rules for the DES algorithm (one required)	
A28MACGN	Specifies to generate a MAC as defined by the AusPayNet standard AS2805.5.4. A Residue Value keyword must be specified.
A28MACVR	Specifies to verify a MAC as defined by the AusPayNet standard AS2805.5.4. A Residue Value keyword must be specified.
A28OWFCL	Specifies to process the value in the chain_data parameter as clear data and return the results of the One Way Function as specified in AS2805.5.4 in the ciphertext parameter.
A28OWFEC	Specifies to process the value in the chain_data parameter as ECB encrypted using the key supplied in the key_parms parameter. The value will be decrypted and the results of the One Way Function as specified in AS2805.5.4 will be returned in the ciphertext parameter.
Rules for the AES algorithm (one optional)	
CBC	Performs encryption in cipher block chaining (CBC) mode. The text length must be a multiple of the AES block size (16-bytes). This is the default value.
ECB	Performs encryption in electronic code book (ECB) mode. The text length must be a multiple of the AES block size (16-bytes).
GCM	Perform Galois/Counter mode encryption. The plaintext may be any length. The ciphertext will have the same length as the plaintext. The key_parms_length and key_parms parameters are used to indicate the length of the tag (the value t) on input and contains the tag on output. Additional Authenticated Data (AAD) is contained in the optional_data_length and optional_data parameters.
PKCS-PAD	Performs encryption in cipher block chaining (CBC) mode, but the data is padded using PKCS padding rules. The length of the clear text data does not have to be a multiple of the cipher block length. The cipher text will be longer than the clear text by at least one byte, and up to 16-bytes. The PKCS padding method is described in “PKCS padding method” on page 1468. This rule should be specified only when there is one request or on the last request of a sequence of chained requests.
X9.23PAD	Performs encryption in cipher block chaining (CBC) mode, but the data is padded using X9.23 padding scheme. The length of the clear text data does not have to be a multiple of the cipher block length. The cipher text will be longer than the clear text by at least one byte,

	and up to 16-bytes. This rule should be specified only when there is one request or on the last request of a sequence of chained requests.
Residue Value (one, required with A28MACVR) *Only valid with processing rules A28MACVR.*	
A28RES	Specifies that there is a residue value in the first 8 bytes of the chain_data parameter. These 8 bytes will be overwritten in the return. See the chain_data parameter for details.
A28NORES	Specifies that there is no residue value in the first 8 bytes of the chain_data parameter. These 8 bytes should be left null, as they will be overwritten in the return. See the chain_data parameter for details. Only valid with A28MACVR.
Key Rule (required, one keyword)	
KEYIDENT	This indicates that the value in the key_identifier parameter is either an internal key token or the label of a key token in the CKDS. The key must be a secure AES key, that is, enciphered under the current master key.
ICV Selection (one, required for GCM, otherwise optional)	
INITIAL	This specifies that this is the first request of a sequence of chained requests and indicates that the initialization vector should be taken from the initialization_vector parameter. This is the default value for CBC and PKDS-PAD. This keyword is not valid with processing rule GCM.
CONTINUE	This specifies that this request is part of a sequence of chained requests, and is not the first request in that sequence. The initialization vector will be taken from the work area identified in the chain_data parameter. This keyword is only valid for processing rules CBC or PKCS-PAD. This keyword is not valid with the ECB or GCM processing rule keyword.
ONLY	Specifies that this is the only request and indicates that the initialization vector should be taken from the initialization_vector parameter. Only valid with the processing rule GCM, A28MACGN, A28MACVR, A28OWFEC, and A28OWFCL. This is the default for A28MACGN, A28MACVR, A28OWFEC, and A28OWFCL.

key_identifier_length

Direction	Type
Input	Integer

The length of the key_identifier parameter in bytes. The length must be 64 bytes for a fixed-length (version X'04') token or a CKDS label, or between the actual length of the token and 725 for a variable-length (version X'05') token.

key_identifier

Direction	Type
Input/Output	String

The identifier of the key to encrypt the text. The key identifier is an operational token or the key label of an operational token in key storage.

For processing rule keywords CBC, ECB, GCM, PKCS-PAD, and X9.23PAD, the key algorithm is AES and the key type must be DATA (fixed-length token, version X'04') or CIPHER (variable-length token, version X'05'). For the CIPHER key, the key usage must

indicate ENCRYPT and the appropriate mode of encryption (CBC, ECB, GCM, or ANY-MODE).

For processing rule keywords A28MACGN and A28MACVR, the key algorithm is DES, the key type is MAC, and the key usages must be ANY-MAC. The key must be a double-length key.

For processing rule keywords A28OWFEC, the key algorithm is DES and the key type is EXPORTER. The key must be a double-length key.

For processing rule keywords A28OWFCL, the key algorithm is DES, the key type is CIPHER, and the key usages must permit decryption. The key must be a double-length key.

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

key_parms_length

Direction	Type
Input	Integer

The length of the key_parms parameter in bytes.

For the GCM processing rule, this is the length of the authentication tag to be verified. Valid lengths are 4, 8, 12, 13, 14, 15, and 16, but using a length of 4 or 8 is strongly discouraged. If there is an error in processing, this value will be set to zero on output. Otherwise, it will be unchanged.

For processing rule keywords A28MACGN, A28MACVR, and A28OWFEC, the value is 64.

For all other processing rules, the value must be zero.

key_parms

Direction	Type
Input/Output	String

The key_parms parameter contains key related parameters.

For the GCM processing rule, key_parms will contain the generated authentication tag for the provided plaintext (plain_text parameter) and additional authenticated data (optional_data parameter). You must specify this generated key_parms when deciphering the text.

For processing rule keywords A28MACGN, A28MACVR, and A28OWFEC, this is the key used to encrypt the chain_data parameter. The key algorithm is DES, the key type is CIPHER. The key must be a double-length key.

When the value of the key_parms_length parameter is 0, this parameter is ignored.

block_size

Direction	Type
Input	Integer

The block size for the cryptographic algorithm. The block size for AES is 16. The block size for DES is 8.

initialization_vector_length

Direction	Type
Input	Integer

The length of the initialization_vector parameter in bytes. For CBC, PKCS-PAD, or X9.23PAD, the length must be equal to the block length for the algorithm specified, 16. For the GCM processing rule, NIST recommends a length of 12, but tolerates any non-zero length up to a maximum of 232-1.

This parameter is ignored when the process rule is ECB.

initialization_vector

Direction	Type
Input	String

This parameter contains the initialization vector (IV) for CBC mode decryption. This includes CBC, GCM, PKCS-PAD, and X9.23PAD processing rule keywords. The same IV value must be used when the data is decrypted.

This parameter is ignored when the process rule is ECB, A28MACGN, A28MACVR, A28OWFEC, and A28OWFCL.

chain_data_length

Direction	Type
Input/Output	Integer

The length of the chain_data parameter in bytes. On input, it contains the length of the buffer provided with parameter chain_data. On output, it is updated with the length of the data returned in the chain_data parameter.

For CBC, PKCS-PAD, and X9.23PAD the value must be at least 32.

For ECB and GCM, this parameter is ignored.

For processing rule keywords A28MACGN and A28MACVR, the value must be 8.

For processing rule keywords A28OWFEC, the value must be 8 or 16.

For processing rule keywords A28OWFCL, the value must be 1 to 16 inclusive.

chain_data

Direction	Type
/Output	String

A buffer that is used as a work area for sequences of chained symmetric algorithm encipher requests. The exact content and layout of chain_data is not described. Your application program must not change the data in this string.

When the keyword INITIAL is used, this is an output parameter and receives data that is needed when enciphering the next part of the input data. When the keyword CONTINUE is used, this is an input/output parameter; the value received as output from the previous

call in the sequence is provided as input to this call, and in turn, this call will return new chain_data that will be used as input on the next call. When CONTINUE is used, both the data (chain_data parameter) and the length (chain_data_length parameter) must be the same values that were received in these parameters as output on the preceding call to the service in the chained sequence.

For processing rule keywords A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, this parameter contains the data that will be processed. When AS28RES is specified, the first 8 bytes will be the residue value from a previous MAC operation. When AS28NORES is specified, the first 8 bytes will be zeros.

A28MACGN, A28MACVR:

Input: (8 bytes)
Input residue value enciphered by the key specified in key_parms parameter or zero.
Output: (8 bytes)
The output residue value enciphered by the key in key_parms parameter.

A28OWFEC:

Input:
Text enciphered by the key specified in key_parms parameter to be used as input to the OWF.
No output in this parameter.

A28OWFCL:

Input:
Clear text that will be used as input to the OWF.
No output in this parameter.

For ECB and GCM, this parameter is ignored.

clear_text_length

Direction	Type
Input	Integer

The length of the clear text data in the clear_text parameter in bytes. For CBC and ECB processing rules, the length must be a multiple of the algorithm block size. For PKDS-PAD and GCM processing rules, the length may be any value. The maximum length is 2^{32}-1.

For GCM, the value may be zero.

When the Crypto Express adapter is a CEX5 or CEX6, the maximum value is 2^{29}-1.
When the Crypto Express adapter is a CEX7, the maximum value is 2^{32}-1.

For processing rules A28OWFEC and A28OWFCL, the value must be zero.
For processing rules A28MACGN and A28MACVR, the maximum length is 1024.

clear_text

Direction	Type
Input	String

The text to be enciphered.

When the clear_text_length is zero, this parameter is ignored.

cipher_text_length

Direction	Type
Input/Output	Integer

On input, this parameter specifies the size of the storage pointed to by the cipher_text parameter. On output, this parameter has the actual length of the text stored in the buffer addressed by the cipher_text parameter.

If process rule PKCS-PAD or X9.23PAD is specified, the cipher text length will exceed the clear text length by at least one byte, and up to 16-bytes. For other process rules, the cipher text length will be equal to the clear text length.

For processing rule keywords A28MACGN and A28MACVR, the value will be 8.

For processing rule keyword A28OWFEC, the value will be 4.

For processing rule keywords A28OWFCL, the value will be the length of the chain_data parameter.

cipher_text

Direction	Type
Input/Output	String

The enciphered text the service returns. If PKCS-PAD, or X9.23PAD is specified, on output the ciphertext buffer contains 1 - 16 bytes of data more than the cleartext input buffer contains.

For processing rule keyword A28MACGN, the 8-byte generated MAC will be returned.

For processing rule keyword A28MACVR, on input, this parameter contains the 8-byte MAC to be verified

For processing rule keywords A28OWFEC and A28OWFCL, the output of the APN OWF will be returned.

optional_data_length

Direction	Type
Input	Integer

The length of the optional_data parameter in bytes. For the GCM processing rule, this parameter contains the length of the Additional Authenticated Data (AAD). The value may be 0 to 2^{32}-1.

For all other processing rules, the value must be 0.

optional_data

Direction	Type
Input/Output	Integer

Optional data required by a specified algorithm or processing mode. For the GCM processing rule, this parameter contains the Additional Authenticated Data (AAD). For all other processing rules, this field is ignored.

You must specify the same optional_data used when deciphering the text.

cipher_text_id

Direction	Type
Input	Integer

For CSNBSAE1 and CSNESAE1 only, the ALET of the dataspace in which the cipher_text parameter resides.

clear_text_id

Direction	Type
Input	Integer

For CSNBSAE1 and CSNESAE1 only, the ALET of the dataspace in which the clear_text parameter resides.

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

The clear_text and cipher_text parameters may be in any dataspace. The initialization_vector and optional_data parameters must be in the caller's address space (primary).

Access control point

The access controls for Symmetric Algorithm Encipher are listed in this table

Access controls for Symmetric Algorithm Encipher	
Rule array keyword	**Access control**
AES	Symmetric Algorithm Encipher - secure AES keys
A28MACGN, A28MACVR	Symmetric Algorithm Encipher – allow A28MACGN and A28MACVR
A28OWFCL	Symmetric Algorithm Encipher - allow APN A28OWFCL
A28OWFEC	Symmetric Algorithm Encipher - allow APN A28OWFEC
GCM	Symmetric Algorithm Encipher – Galois/Counter mode AES

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 49. Symmetric Algorithm Encipher required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	Secure AES key support requires the Nov. 2008 or later licensed internal code (LIC). Keywords GCM and ONLY are not supported.

		Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor Crypto Express3 Coprocessor	Secure AES key support requires the Nov. 2008 or later licensed internal code (LIC). Keywords GCM and ONLY are not supported. Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	AES Variable-length Symmetric Internal Key Tokens require the Sep. 2011 or later licensed internal code (LIC). Keywords GCM and ONLY are not supported. Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Keywords GCM and ONLY are not supported. Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.

	Crypto Express6 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Rule array keyword X9.23PAD requires the CCA release 6.7 or later licensed internal code (LIC). Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, and A28NORES are not supported.
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Keywords GCM and ONLY require the March 2016 or later licensed internal code (LIC). Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Rule array keyword X9.23PAD requires the CCA release 6.7 or later licensed internal code (LIC). Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, and A28NORES are not supported.
	Crypto Express7 CCA Coprocessor	Rule array keywords DES, A28MACGN, A28MACVR, A28OWFEC, A28OWFCL, A28RES, A28NORES, and X9.23PAD require the CCA release 7.4 or later licensed internal code (LIC).

Enhanced PIN security

Enhanced PIN security mode

An Enhanced PIN Security Mode is available. This optional mode is selected by enabling the **Enhanced PIN Security** access control point in coprocessor domain role. When active, this control point affects all PIN callable services that extract or format a PIN using a PIN-block format of 3621 or 3624 with a PIN-extraction method of PADDIGIT.

Table 50 summarizes the callable services affected by the Enhanced PIN Security Mode and describes the effect that the mode has when the access control point is enabled.

Table 50. Callable services affected by enhanced PIN security mode

PIN-block format and PIN-extraction method	Callable services affected	PIN processing changes when Enhanced PIN Security Mode enabled
ECI-2, 3621, or 3624 formats AND PINLENxx	CSNBCPA CSNBPTR CSNBPTRE CSNBPVR CSNBPVR2	The PINLENxx keyword in rule_array parameter for PIN extraction method is not allowed if the Enhanced PIN Security Mode is enabled. Note: The services will fail with return

		code 8 reason code '7E0'x.
3621 or 3624 format and PADDIGIT	CSNBCPA CSNBPTR CSNBPTR2 CSNBPTRE CSNBPVR CSNBPVR2 CSNBPCU	PIN extraction determines the PIN length by scanning from right to left until a digit, not equal to the pad digit, is found. The minimum PIN length is set at four digits, so scanning ceases one digit past the position of the 4th PIN digit in the block.
3621 or 3624 format and PADDIGIT	CSNBCPE CSNBEPG CSNBPTR CSNBPTR2 CSNBPTRE	PIN formatting does not examine the PIN, in the output PIN block, to see if it contains the pad digit.
3621 or 3624 format and PADDIGIT	CSNBPTR CSNBPTR2 CSNBPTRE	Restricted to non-decimal digit for PAD digit.

Enhanced PIN checking for CSNBPTR and CSNBPTR2

For the PIN translate services, additional checking is available when the TRANSLAT rule array keyword is specified. When the **Encrypted PIN Translate - Translate PIN Check** access control is enabled, checking of the PIN block is performed. The checking is similar to the checking done when the REFORMAT keyword is specified.

PIN block error processing mode

To prevent the abuse of PIN processing error messages, due to information leakage derived from the return code reason codes returned under various conditions, the PIN checking errors will be replaced with a general ISO format error. This optional mode is selected by enabling the **General ISO PIN Error Mode** access control in the coprocessor domain role. These services are affected:

- Encrypted PIN Translate (CSNBPTR and CSNEPTR)
- Encrypted PIN Translate2 (CSNBPTR2 and CSNEPTR2)
- DK PIN Change (CSNBDPC and CSNEDPC)
- DK PIN Verify (CSNBDPV and CSNEDPV)

Return code 8 reason code 2514 (9D2) will be issued instead of these return code 8 reason codes: 100, 106, 110, 108, 407, 3004, 3016

Encrypted PIN Translate (CSNBPTR and CSNEPTR)

Access control points

The following table shows the access control points in the domain role that control the function of this service.

Table 51. Required access control points for Encrypted PIN Translate

Processing rule	Access control point
TRANSLAT	Encrypted PIN Translate - Translate
REFORMAT	Encrypted PIN Translate - Reformat

If any of the Unique Key per Transaction rule array keywords are specified, the **DUKPT - PIN Verify, PIN Translate** access control point must be enabled.

An enhanced PIN security mode is available for extracting PINs from a 3621 or 3624 encrypted PIN-block and formatting an encrypted PIN block into IBM 3621 or 3624 format using the PADDIGIT PIN-extraction method. This mode limits checking of the PIN to decimal digits, and a minimum PIN length of 4 is enforced; no other PIN-block consistency checking will occur. To activate this mode, enable the **Enhanced PIN Security** access control.

When the **Encrypted PIN Translate - Translate PIN Check** access control is enabled, checking of the PIN block is performed. The checking is like the checking done when the REFORMAT keyword is specified.

When the **General ISO PIN Error Mode** access control is enabled, the return code will be a general PIN block error (8/2514) instead of some of the PIN block errors return code. The use of a general return code can prevent the abuse of PIN processing error messages due to information leakage derived from the return code reason codes returned under various conditions. See ‘PIN block error processing mode’ on page 80 for details.

Three additional access controls should be considered: **ANSI X9.8 PIN - Enforce PIN block restrictions**, **ANSI X9.8 PIN - Allow modification of PAN**, and **ANSI X9.8 PIN - Allow only ANSI PIN blocks**. These three access controls affect how PIN processing is performed as described below. The access controls will affect this and other PIN processing services if enabled.

1. Enable the ANSI X9.8 PIN - Enforce PIN block restrictions access control to apply additional restrictions to PIN processing as follows:
   - Do not translate or reformat a non-ISO PIN block into an ISO PIN block. Specifically, do not allow an IBM 3624 PIN-block format in the output_PIN_profile variable when the PIN-block format in the input_PIN_profile variable is not IBM 3624.
   - Constrain use of ISO-2 PIN blocks to offline PIN verification and PIN change operations in integrated circuit card environments only. Specifically, do not allow ISO-2 input or output PIN blocks.
   - Do not translate or reformat a PIN-block format that includes a PAN into a PIN-block format that does not include a PAN. Specifically, do not allow an ISO-1 PIN-block format in the output_PIN_profile variable when the PIN-block format in the input_PIN_profile variable is ISO-0, ISO-3, or ISO-4.
   - Do not allow a change of PAN data. Specifically, when performing translations between PIN block formats that both include PAN data, do not allow the input_PAN_data and output_PAN_data variables to be different from the PAN data enciphered in the input PIN-block.
2. Enable the **ANSI X9.8 PIN - Allow modification of PAN** access control to override the restriction to not allow a change of PAN data. This override is applicable only when either the **ANSI X9.8 PIN - Enforce PIN block restrictions** control, the **ANSI X9.8 PIN - Allow only ANSI PIN blocks** control, or both are enabled. This override is to support account number changes in issuing environments. The **ANSI X9.8 PIN - Allow modification of PAN** control has no effect if neither **the ANSI X9.8 PIN - Enforce PIN block restrictions** control nor the **ANSI X9.8 PIN - Allow only ANSI PIN blocks** control is enabled. This rule does not apply for CSNBPTRE, and PAN changes are not allowed.
3. Enable the ANSI X9.8 PIN - Allow only ANSI PIN blocks control to apply a more restrictive variation of the **ANSI X9.8 PIN - Enforce PIN block restrictions** control. In addition to the previously described restrictions of the **ANSI X9.8 PIN - Enforce PIN block restrictions** control, this control also restricts the input_PIN_profile and the output_PIN_profile to contain only ISO-0, ISO-1, ISO-3, and ISO-4 PIN block formats. Specifically, the IBM 3624 PIN-block

format is not allowed with this command. The **ANSI X9.8 PIN - Allow only ANSI PIN blocks** control overrides the **ANSI X9.8 PIN - Enforce PIN block restrictions** control.

When the **Prohibit translation from DES wrapping to weaker DES wrapping** access control point is enabled in the domain role, this service will fail if the input_PIN_encrypting_key_identifier is stronger than the output_PIN_encrypting_key_identifier.

When the **Disallow PIN block format ISO-1** access control is enabled in the domain role, the PIN block format in the input_PIN_profile and output_PIN_profile parameters is not allowed to be ISO-1.

Encrypted PIN Translate2 (CSNBPTR2 and CSNEPTR2)

Access control points

The following table shows the access control points in the domain role that control the function of this service. When the input or output PIN format in the PIN profile is ISO-4, the **Encrypted PIN Translate2 – REFORMAT/TRANSLATE** access controls are used. When neither the input nor output PIN format in the PIN profile is ISO-4, the **Encrypted PIN Translate – REFORMAT/TRANSLATE** access controls are used.

Table 52. Required access control points for Encrypted PIN Translate2

Processing rule	Access control point
TRANSLAT	• Encrypted PIN Translate - TRANSLAT • Encrypted PIN Translate2 - TRANSLAT
REFORMAT	• Encrypted PIN Translate - REFORMAT • Encrypted PIN Translate2 - REFORMAT

Table 53. Required access controls for ISO-4 PIN blocks

Input PIN format	Output PIN format	Authenticated PAN-change allowed	Access control name
ISO-0	ISO-4	No	Encrypted PIN Translate2 – Permit ISO-0 to ISO-4 Reformat.
ISO-1	ISO-4	No	Encrypted PIN Translate2 – Permit ISO-1 to ISO-4 Reformat (see note 1).
ISO-1	ISO-4	No	Encrypted PIN Translate2 – Permit ISO-1 to ISO-4 RFMT1TO4 (see note 1).
ISO-4	ISO-0	No	Encrypted PIN Translate2 – Permit ISO-4 to ISO-0 Reformat.
ISO-4	ISO-1	No	Encrypted PIN Translate2 – Permit ISO-4 to ISO-1 Reformat (see note 2).
ISO-4	ISO-1	No	Encrypted PIN Translate2 – Permit ISO-4 to ISO-1 RFMT4TO1 (see note 2).
ISO-4	ISO-4	No	Encrypted PIN Translate2 – Permit ISO-4 to ISO-4 Translate.
ISO-4	ISO-4	Yes	Encrypted PIN Translate2 – Permit ISO-4 Reformat with PAN Change (see note 3).

ISO-4	ISO-4	Yes	Encrypted PIN Translate2 - Permit ISO-4 to ISO-4 PTR2AUTH (see note 3).

Notes:

1. When enabled, the **Encrypted PIN Translate2 – Permit ISO-1 to ISO-4 RFMT1TO4** control has the effect of disallowing REFORMAT requests from ISO-1 to ISO-4 PIN blocks unless the outbound PIN encrypting key has the RFMT1TO4 key-usage field bit enabled in the AES key-token.
2. When enabled, the **Encrypted PIN Translate2 – Permit ISO-4 to ISO-1 RFMT4TO1** control has the effect of disallowing REFORMAT requests from ISO-4 to ISO-1 PIN blocks unless the inbound PIN encrypting key has the RFMT4TO1 key-usage field bit enabled in the AES key-token.
3. When enabled, the **Encrypted PIN Translate2 – Permit ISO-4 to ISO-4 PTR2AUTH** control has the effect of disallowing REFORMAT requests from ISO-4 to ISO-4 PIN blocks unless the outbound PIN encrypting key has the PTR2AUTH key-usage field bit enabled in the AES key-token.

If any of the Unique Key per Transaction rule array keywords are specified, the **DUKPT - PIN Verify, PIN Translate** access control point must be enabled.

An enhanced PIN security mode is available for extracting PINs from a 3621 or 3624 encrypted PINblock and formatting an encrypted PIN block into IBM 3621 or 3624 format using the PADDIGIT PINextraction method. This mode limits checking of the PIN to decimal digits, and a minimum PIN length of 4 is enforced; no other PIN-block consistency checking will occur. To activate this mode, enable the **Enhanced PIN Security** access control.

When the **Encrypted PIN Translate - Translate PIN Check** access control is enabled, checking of the PIN block is performed. The checking is like the checking done when the REFORMAT keyword is specified.

When the **General ISO PIN Error Mode** access control is enabled, the return code will be a general PIN block error (8/2514) instead of some of the PIN block errors return code. The use of a general return code can prevent the abuse of PIN processing error messages due to information leakage derived from the return code reason codes returned under various conditions. See ‘PIN block error processing mode’ on page 80 for details.

Three additional access controls should be considered: **ANSI X9.8 PIN - Enforce PIN block restrictions**, **ANSI X9.8 PIN - Allow modification of PAN**, and **ANSI X9.8 PIN - Allow only ANSI PIN blocks**. These three access controls affect how PIN processing is performed as described below. The access controls will affect this and other PIN processing services if enabled.

1. Enable the **ANSI X9.8 PIN - Enforce PIN block restrictions** access control to apply additional restrictions to PIN processing as follows:
   - Do not translate or reformat a non-ISO PIN block into an ISO PIN block. Specifically, do not allow an IBM 3624 PIN-block format in the output_PIN_profile variable when the PIN-block format in the input_PIN_profile variable is not IBM 3624.
   - Constrain use of ISO-2 PIN blocks to offline PIN verification and PIN change operations in integrated circuit card environments only. Specifically, do not allow ISO-2 input or output PIN blocks.
   - Do not translate or reformat a PIN-block format that includes a PAN into a PIN-block format that does not include a PAN. Specifically, do not allow an ISO-1 PIN-block

format in the output_PIN_profile variable when the PIN-block format in the input_PIN_profile variable is ISO-0, ISO-3, or ISO-4.
    - Do not allow a change of PAN data. Specifically, when performing translations between PIN block formats that both include PAN data, do not allow the input_PAN_data and output_PAN_data variables to be different from the PAN data enciphered in the input PIN-block.
2. Enable the **ANSI X9.8 PIN - Allow modification of PAN** access control to override the restriction to not allow a change of PAN data. This override is applicable only when either the **ANSI X9.8 PIN - Enforce PIN block restrictions** control, the **ANSI X9.8 PIN - Allow only ANSI PIN blocks** control, or both are enabled. This override is to support account number changes in issuing environments. The **ANSI X9.8 PIN - Allow modification of PAN** control has no effect if neither the ANSI X9.8 PIN - Enforce PIN block restrictions control nor **the ANSI X9.8 PIN - Allow only ANSI PIN blocks** control is enabled. This rule does not apply for CSNBPTRE, and PAN changes are not allowed.
3. Enable the **ANSI X9.8 PIN - Allow only ANSI PIN blocks** control to apply a more restrictive variation of the **ANSI X9.8 PIN - Enforce PIN block restrictions** control. In addition to the previously described restrictions of the **ANSI X9.8 PIN - Enforce PIN block restrictions** control, this control also restricts the input_PIN_profile and the output_PIN_profile to contain only ISO-0, ISO-1, ISO-3, and ISO-4 PIN block formats. Specifically, the IBM 3624 PIN-block format is not allowed with this command. The **ANSI X9.8 PIN - Allow only ANSI PIN blocks** control overrides the **ANSI X9.8 PIN - Enforce PIN block restrictions** control.

When the **Disallow translation from AES wrapping to DES wrapping** access control point is enabled in the domain role, this service fails if the input_PIN_encrypting_key_identifier is an AES key and the output_PIN_encrypting_key_identifier is a DES key.

When the **Disallow translation from AES wrapping to weaker AES wrapping** access control point is enabled in the domain role, this service fails if the input_PIN_encrypting_key_identifier is stronger than the output_PIN_encrypting_key_identifier.

When the **Disallow translation from DES wrapping to weaker DES wrapping** access control point is enabled in the domain role, this service fails if the input_PIN_encrypting_key_identifier is stronger than the output_PIN_encrypting_key_identifier.

When the **Disallow PIN block format ISO-1** access control is enabled in the domain role, the PIN block format in the input_PIN_profile and output_PIN_profile parameters is not allowed to be ISO-1.

Encrypted PIN Verify2 (CSNBPVR2 and CSNEPVR2)

The Encrypted PIN Verify2 callable service validates a customer encrypted PIN-block against a reference encrypted PIN block.

You can specify these PIN-block formats:
- IBM 3624
- ISO-0 (same as ANS X9.8, VISA-1, and ECI-1)
- ISO-1 (same as ECI-4)
- ISO-2
- ISO-3
- ISO-4

The service supports truncated customer PINs, optionally verifying an indicated number of PIN digits (minimum of 4) that is less than the number of digits for the reference PIN. When truncated PINs are compared, the order is right to left for the number of digits specified in the PIN_check_length parameter.

The derived unique-key-per-transaction (DUKPT) algorithm is available. Both DES-DUKPT (ANSI X9.24-1 2007) and AES-DUKPT (ANSI x9.24-3 2017) are supported. This support is available for the input_PIN_encrypting_key_identifier parameter and the reference_PIN_encrypting_key_identifier parameter.

The callable service name for AMODE(64) invocation is CSNEPVR2.

Format

```
CALL CSNBPVR2(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              reference_PIN_rule_array_count,
              reference_PIN_rule_array,
              PIN_check_length,
              input_PIN_encrypting_key_identifier_length,
              input_PIN_encrypting_key_identifier,
              reference_PIN_encrypting_key_identifier_length,
              reference_PIN_encrypting_key_identifier,
              input_PIN_profile_length,
              input_PIN_profile,
              input_PIN_block_length,
              input_PIN_block,
              reference_PIN_profile_length,
              reference_PIN_profile,
              reference_PIN_block_length,
              reference_PIN_block,
              input_PAN_data,
              reference_PAN_data,
              reserved1_length,
              reserved1,
              reserved2_length,
              reserved2,
              reserved3_length,
              reserved3,
              reserved4_length,
              reserved4 )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1359 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1359 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count,

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value must be 1, 2, or 3.

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 54. Keyword for Encrypted PIN Verify2

Keyword	Meaning
Processing rule (one required)	
REFPIN	Specifies that the input PIN is to be compared to the reference PIN.
TRUNCPIN	Specifies that the input PIN is to be compared to a truncated version of the reference PIN for the number of digits specified by the

	PIN_check_length parameter. Note: The digits of the PINs are checked from the rightmost digit to the left for the number of digits specified.
Input PIN unique key per transaction (one, optional).	
UKPT	Specifies the use of the single-DES method of DUKPT key derivation and PIN-block decryption for the input PIN encrypting key.
DUKPT	Specifies the use of the triple-DES method of DUKPT key derivation and PIN-block decryption for the input PIN encrypting key.
ADUKPT	Specifies the use of the AES DUKPT method of DUKPT key-derivation and PIN-block decryption for the input PIN encrypting key.
Input PIN-extraction method (one, optional). See “PIN block format and PIN extraction method keywords” on page 675 for additional information and a list of PIN block formats and PIN extraction method keywords. **Note**: If a PIN extraction method is not specified, the first one listed in Table 264 on page 675 for the PIN block format will be the default.	

reference_PIN_rule_array_count,

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value must be 0, 1, or 2.

reference_PIN_rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 55. Keywords for Encrypted PIN Verify2

Keyword	Meaning
Reference PIN unique key per transaction (one, optional).	
UKPT	Specifies the use of the single-DES method of DUKPT key derivation and PIN-block decryption for the reference PIN encrypting key.
DUKPT	Specifies the use of the triple-DES method of DUKPT key derivation and PIN-block decryption for the reference PIN encrypting key.
ADUKPT	Specifies the use of the AES DUKPT method of DUKPT key-derivation and PIN-block decryption for the reference PIN encrypting key.
Reference PIN-extraction method (one, optional). See “PIN block format and PIN extraction method keywords” on page 675 for additional information and a list of PIN block formats and PIN extraction method keywords.	

Note: If a PIN extraction method is not specified, the first one listed in Table 264 on page 675 for the PIN block format will be the default.

.**PIN_check_length**

Direction	Type
Input	Integer

The number of digits of the reference PIN to compare when the TRUNCPIN keyword is specified. The value may be 4-16 inclusive and must be less or equal to the length of the reference PIN. When the REFPIN keyword is specified, the value must be zero.

input_PIN_encrypting_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *input_PIN_encrypting_key_identifier* parameter. If the *input_PIN_encrypting_key_identifier* contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

input_PIN_encrypting_key_identifier

Direction	Type
Input	String

This is either the identifier of the key to unwrap the input PIN block or the identifier of the key-generating key used to derive the key to unwrap the input PIN block. The key identifier is an operational token or the key label of an operational token in key storage.

The key identifier must identify an AES key when the input PIN profile specifies a PIN-block format of ISO-4, otherwise it must identify a DES key.

For DES keys, the control vector in the key token must specify the IPINENC key-type with EPINVER bit (CV bit 19) set to B'1'.

For AES keys, the variable-length symmetric key token must have a token algorithm of AES and a key type of PINPROT. In addition, the key usage fields must indicate that the key can be used for decryption (DECRYPT), the encryption mode must be Cipher Block Chaining (CBC), common usage control must be NOFLDFMT, PIN block format usage must be ISO-4, and PIN function usage EPINVER must be enabled.

When any of the DUKPT rule array keywords is used for the input PIN encrypting key:

- When you use the DES DUKPT process for the input PIN-block, specify that the base derivation key as a KEYGENKY key type with the UKPT bit (CV bit 18) set to B'1'.
- When you use the AES DUKPT process for the input PIN-block, specify the base derivation key as an AES DKYGENKY key with the A-DUKPT bit set to 1 in the low-order byte of key usage field 1.

When the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

reference_PIN_encrypting_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *reference_PIN_encrypting_key_identifier* parameter. If the *reference_PIN_encrypting_key_identifier* contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

reference_PIN_encrypting_key_identifier

Direction	Type
Input	String

This is either the identifier of the key to unwrap the reference PIN block or the identifier of the key-generating key used to derive the key to unwrap the reference PIN block. The key identifier is an operational token or the key label of an operational token in key storage.

The key identifier must identify an AES key when the reference PIN profile specifies a PIN-block format of ISO-4, otherwise it must identify a DES key.

For DES keys, the control vector in the key token must specify the IPINENC key-type with EPINVER bit (CV bit 19) set to B'1'.

For AES keys, the variable-length symmetric key token must have a token algorithm of AES and a key type of PINPROT. In addition, the key usage fields must indicate that the key can be used for decryption (DECRYPT), the encryption mode must be Cipher Block Chaining (CBC), common usage control must be NOFLDFMT, PIN block format usage must be ISO-4, and PIN function usage EPINVER must be enabled.

When any of the DUKPT rule array keywords is used for the reference PIN encrypting key:

- When you use the DES DUKPT process for the reference PIN-block, specify that the base derivation key as a KEYGENKY key type with the UKPT bit (CV bit 18) set to B'1'.
- When you use the AES DUKPT process for the reference PIN-block, specify the base derivation key as an AES DKYGENKY key with the A-DUKPT bit set to 1 in the low-order byte of key usage field 1.

When the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

input_PIN_profile_length

Direction	Type
Input	Integer

The length of the input_PIN_profile parameter in bytes.

Table 56. Supported Encrypted PIN Verify2 input PIN profile lengths

Pin profile	Length
PIN-block format only.	24
PIN-block format and CKSN extension used for DES-DUKPT.	48

PIN-block format and single block of derivation data extension used for AES-DUKPT.	44

input_PIN_profile

Direction	Type
Input	String

The three 8-byte character elements that contain information necessary to extract the PIN from a formatted PIN block. See “The PIN profile” on page 674 for additional information.

When the DUKPT keywords are specified for the input PIN encrypting key, additional bytes must be present containing the CKSN or derivation data extension. The DES DUKPT algorithm will be used to derive the DUKPT key used to decrypt the input PIN block when the CKSN extension is included in the input_PIN_profile. The AES DUKPT algorithm will be used to derive the DUKPT key used to decrypt the input PIN block when the derivation data extension is included in the input_PIN_profile. See Table 639 on page 1613 for the layout of the AES-DUKPT derivation data extension. The algorithm indicator must be set to either X'0000' (2-key TDES) or X'0001' (3-key TDES). The key usage indicator must be set to X'1000' (PIN Encryption).

input_PIN_block_length

Direction	Type
Input	Integer

The length of the input_PIN_block in bytes. The value must be 16 for ISO-4 PIN blocks and 8 for all other PIN blocks.

input_PIN_block

Direction	Type
Input	String

The encrypted PIN block containing the PIN to compare against the reference PIN.

reference_PIN_profile_length

Direction	Type
Input	Integer

The length of the reference_PIN_profile parameter in bytes.

Table 57. Supported Encrypted PIN Verify2 reference PIN profile lengths

Pin profile	Length
PIN-block format only.	24
PIN-block format and CKSN extension used for DES-DUKPT.	48
PIN-block format and single block of derivation data extension used for AES-DUKPT.	44

reference_PIN_profile

Direction	Type
Input	String

The three 8-byte character elements that contain information necessary to extract the reference PIN from a formatted PIN block. See “The PIN profile” on page 674 for additional information.

When the DUKPT keywords are specified for the reference PIN encrypting key, additional bytes must be present containing the CKSN or derivation data extension. The DES DUKPT algorithm will be used to derive the DUKPT key used to decrypt the input PIN block when the CKSN extension is included in the reference_PIN_profile. The AES DUKPT algorithm will be used to derive the DUKPT key used to decrypt the input PIN block when the derivation data extension is included in the reference_PIN_profile. See Table 639 on page 1613 for the layout of the AES-DUKPT derivation data extension. The algorithm indicator must be set to either X'0000' (2-key TDES) or X'0001' (3-key TDES). The key usage indicator must be set to X'1000' (PIN Encryption).

reference_PIN_block_length

Direction	Type
Input	Integer

The length of the reference_PIN_block in bytes. The value must be 16 for ISO-4 PIN blocks and 8 for all other PIN blocks.

reference_PIN_block

Direction	Type
Input	String

The encrypted PIN block containing the reference PIN.

input_PAN_data

Direction	Type
Input	String

The PAN data for the input_PIN_block. The PAN data is used if the input PIN profile specifies the ISO-0, ISO-3, ISO-4, or VISA-4 keyword for the PIN block format. The PAN_data parameter is 21 bytes long.

When using the ISO-0, IS0-3, or VISA-4 keyword, the value is 12 bytes long. Use the 12 rightmost digits of the PAN data, excluding the check digit.

When using the ISO-4 keyword, the value is 21 bytes long. The PAN data is 10 –19 bytes long. The length of the PAN data and the PAN data are contained in the structure below padded to 21 bytes with characters that will be ignored.

Offset	Length	Description
0	2	Length of the PAN data field, p
2	p	10 – 19 bytes of PAN data
2+p	0-9	Padding

Note: If the reference_PAN_data and the input_PAN_data are different lengths, the rightmost 12

digits must be the same excluding the check digit.

reference_PAN_data

Direction	Type
Input	String

The PAN data for the reference_PIN_block. The PAN data is used if the reference PIN profile specifies the ISO-0, ISO-3, ISO-4, or VISA-4 keyword for the PIN block format. The PAN_data parameter is 21 bytes long.

When using the ISO-0, IS0-3, or VISA-4 keyword, the value is 12 bytes long. Use the 12 rightmost digits of the PAN data, excluding the check digit.

When using the ISO-4 keyword, the value is 21 bytes long. The PAN data is 10 –19 bytes long. The length of the PAN data and the PAN data are contained in the structure below padded to 21 bytes with characters that will be ignored.

Offset	Length	Description
0	2	Length of the PAN data field, p
2	p	10 – 19 bytes of PAN data
2+p	0-9	Padding

Note: If the reference_PAN_data and the input_PAN_data are different lengths, the rightmost 12 digits must be the same excluding the check digit.

reserved1_length

Direction	Type
Input	Integer

Length in bytes of the reserved1 parameter. The value must be 0.

reserved1

Direction	Type
Input	String

This parameter is ignored.

reserved2_length

Direction	Type
Input	Integer

Length in bytes of the reserved2 parameter. The value must be 0.

reserved2

Direction	Type
Input	String

This parameter is ignored.

reserved3_length

Direction	Type
Input	Integer

Length in bytes of the reserved3 parameter. The value must be 0.

reserved3

Direction	Type
Input	String

This parameter is ignored.

reserved4_length

Direction	Type
Input	Integer

Length in bytes of the reserved4 parameter. The value must be 0.

reserved4

Direction	Type
Input	String

This parameter is ignored.

Usage Notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

Access control points

This table shows the access control points in the domain role that control the function of this service.

Table 57. Required access controls for Encrypted PIN Verify2

Processing rule	Access control
REFPIN	Encrypted PIN Verify2 - REFPIN
TRUNCPIN	Encrypted PIN Verify2 - TRUNCPIN

If any of the Unique Key per Transaction rule array keywords are specified, the **DUKPT - PIN Verify, PIN Translate** access control must be enabled.

An enhanced PIN security mode is available for extracting PINs from a 3621 or 3624 encrypted PIN-

block and formatting an encrypted PIN block into IBM 3621 or 3624 format using the PADDIGIT PIN-extraction method. This mode limits checking of the PIN to decimal digits, and a minimum PIN length of 4 is enforced; no other PIN-block consistency checking will occur. To activate this mode, enable the **Enhanced PIN Security** access control.

If the **ANSI X9.8 PIN – Use stored decimalization tables only** access control is enabled in the domain role, any decimalization table specified must match one of the active decimalization tables in the coprocessors.

When the **Disallow PIN block format ISO-1** access control is enabled in the domain role, the PIN block format in the input_PIN_profile parameter is not allowed to be ISO-1.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 58. Encrypted PIN Verify2 required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC		This service is not supported.
IBM System z10 EC IBM System z10 BC		This service is not supported.
IBM zEnterprise 196 IBM zEnterprise 114		This service is not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12		This service is not supported.
IBM z13 IBM z13s		This service is not supported.
IBM z14 IBM z14 ZR1		This service is not supported.
IBM z15 IBM z15 T02	Crypto Express5 CCA Coprocessor Crypto Express6 CCA Coprocessor	This service is not supported.
	Crypto Express7 CCA Coprocessor	This service requires the CCA release 7.4 or later licensed internal code (LIC).

PIN Change/Unblock (CSNBPCU and CSNEPCU)

~~Restrictions~~

~~Triple-length DES keys are not supported.~~

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

Triple-length DES keys requires the CCA release 6.7, 7.4, or later licensed internal code.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 59. PIN Change/Unblock hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	ISO-3 PIN block format requires the Nov. 2007 or later licensed internal code (LIC). AMEXPCU1 and AMEXPCU2 keywords require May, 2012 or later version of LIC. Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor Crypto Express3 Coprocessor	ISO-3 PIN block format requires the Nov. 2007 or later licensed internal code (LIC). AMEXPCU1 and AMEXPCU2 keywords require May, 2012 or later version of LIC for Crypto Express2. AMEXPCU1 and AMEXPCU2 keywords require June, 2012 or later version of LIC for Crypto Express3. Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported Triple-length DES keys are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	AMEXPCU1 and AMEXPCU2 keywords require June, 2012 or later version of LIC. Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	AMEXPCU1 and AMEXPCU2 keywords require June, 2012 or later version of LIC for Crypto Express3. AMEXPCU1 and AMEXPCU2 keywords require September, 2012 or later version of LIC for Crypto Express4.

		Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
	Crypto Express6 CCA Coprocessor	Compliant-tagged key tokens require the July 2019 or later licensed internal code (LIC). PIN block format ISO-4 requires the October 2020 or later licensed internal code (LIC). Triple-length DES keys require the CCA release 6.7 or later licensed internal code (LIC).
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
	Crypto Express6 CCA Coprocessor	PIN block format ISO-4 requires the September 2020 or later licensed internal code (LIC). Triple-length DES keys require the CCA release 6.7 or later licensed internal code (LIC).
	Crypto Express7 CCA Coprocessor	Triple-length DES keys require the CCA release 7.4 or later licensed internal code (LIC).

Secure Messaging for PINs (CSNBSPN and CSNESPN)

~~Restrictions~~

~~Triple-length DES keys are not supported.~~

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

SAF will be invoked to check authorization to use the Secure Messaging for PINs service and any key labels specified as input.

Keys only appear in the clear within the secure boundary of the cryptographic coprocessors and never in host storage.

Triple-length DES keys requires the CCA release 6.7, 7.4, or later licensed internal code.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 60. Secure Messaging for PINs required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	ISO-3 PIN block format requires the Nov. 2007 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor Crypto Express3 Coprocessor	ISO-3 PIN block format requires the Nov. 2007 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported Triple-length DES keys are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor	Compliant-tagged key tokens are not supported.

	Crypto Express4 CCA Coprocessor	PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
	Crypto Express6 CCA Coprocessor	Compliant-tagged key tokens require the July 2019 or later licensed internal code (LIC). PIN block format ISO-4 requires the October 2020 or later licensed internal code (LIC). Triple-length DES keys require the CCA release 6.7 or later licensed internal code (LIC).
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Crypto Express5 CCA Coprocessor Compliant-tagged key tokens are not supported. PIN block format ISO-4 is not supported. Triple-length DES keys are not supported.
	Crypto Express6 CCA Coprocessor	PIN block format ISO-4 requires the September 2020 or later licensed internal code (LIC). Triple-length DES keys require the CCA release 6.7 or later licensed internal code (LIC).
	Crypto Express7 CCA Coprocessor	Triple-length DES keys require the CCA release 7.4 or later licensed internal code (LIC).

SET Block Decompose (CSNDSBD and CSNFSBD)

Usage notes

Triple-length DES keys requires the CCA release 6.7, 7.4, or later licensed internal code.

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

When the SET Block Decompose service is invoked without the DES-ONLY keyword, the DES key is retrieved from the RSA-OAEP block and returned in the key token contained in the DES_key_block. On subsequent calls to the SET Block Decompose service, a caller can re-use the DES key. The caller of the service must supply the DES_key_block, the DES_key_block_length, the DES_encrypted_data_block, the DES_encrypted_data_block_length, the initialization and chaining vectors, and the rule_array keywords SET1.00 and DES-ONLY. The RSA private key information, RSA-OAEP block and length, XData string and length, and hash block and length need not be supplied (although the parameters must still be specified). For this invocation, the decryption of the RSA-OAEP block is bypassed; only DES decryption is performed, using the supplied DES key.

When the SET Block Decompose service is invoked with the PINBLOCK keyword, DES-ONLY may not also be specified. If both of these rule array keywords are specified, the service will fail. If PINBLOCK is specified and the DES_key_block_length field is not 128, the service will fail.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 61. SET Block Decompose required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	Triple-length DES keys are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor Crypto Express3 Coprocessor	Triple-length DES keys are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	Triple-length DES keys are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Triple-length DES keys are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Triple-length DES keys are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Triple-length DES keys are not supported.
	Crypto Express6 CCA Coprocessor	Triple-length DES keys require the CCA release 6.7 or later licensed internal code (LIC).

IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Triple-length DES keys are not supported.
	Crypto Express6 CCA Coprocessor	Triple-length DES keys require the CCA release 6.7 or later licensed internal code (LIC).
	Crypto Express7 CCA Coprocessor	Triple-length DES keys require the CCA release 7.4 or later licensed internal code (LIC).

DK PIN Change (CSNBDPC and CSNEDPC)

Use the DK PIN Change callable service to allow a customer to change their PIN to a value of their choosing.

The current and new PINs are entered into the ATM, where they are encrypted into ISO-1 or ISO-4 PIN blocks. The PIN and other needed information are used to verify the current PIN. If the PIN does not verify, the process is aborted. If the PIN does verify, the PIN is reformatted into a PBF-O format and the provided information is used to create a new PIN reference value.

Note: Regarding weak PINs, if the new PIN specified appears in the weak PIN table, the PIN change fails with an indication that the selected new PIN was not valid.

The callable service name for AMODE(64) invocation is CSNEDPC.

Format

```
CALL CSNBDPC(
       return_code,
       reason_code,
       exit_data_length,
       exit_data,
       rule_array_count,
       rule_array,
       PAN_data_length,
       PAN_data,
       card_p_data_length,
       card_p_data,
       card_t_data_length,
       card_t_data,
       cur_ISO_PIN_block_length,
       cur_ISO_PIN_block,
       new_ISO_PIN_block_length,
       new_ISO_PIN_block,
       card_script_data_length,
       card_script_data,
       script_offset,
       script_offset_field_length,
       script_initialization_vector_length,
       script_initialization_vector,
       output_PIN_profile,
       PIN_reference_value_length,
```

```
PIN_reference_value,
PRW_random_number_length,
PRW_random_number,
PRW_key_identifier_length,
PRW_key_identifier,
cur_IPIN_encryption_key_identifier_length,
cur_IPIN_encryption_key_identifier,
new_IPIN_encryption_key_identifier_length,
new_IPIN_encryption_key_identifier,
script_key_identifier_length,
script_key_identifier,
script_MAC_key_identifier_length,
script_MAC_key_identifier,
new_PRW_key_identifier_length,
new_PRW_key_identifier,
OPIN_encryption_key_identifier_length,
OPIN_encryption_key_identifier,
OEPB_MAC_key_identifier_length,
OEPB_MAC_key_identifier,
script_length,
script,
script_MAC_length,
script_MAC,
new_PIN_reference_value_length,
new_PIN_reference_value,
new_PRW_random_number_length,
new_PRW_random_number,
output_encrypted_PIN_block_length,
output_encrypted_PIN_block,
PIN_block_MAC_length,
PIN_block_MAC)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value must be 0 - 7.

rule_array

Direction	Type
Input	Character

Keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 62. Rule array keywords for the DK PIN Change Service

Keyword	Meaning
PIN Block output selection keyword (One, optional)	
NOEPB	Do not return an encrypted PIN block (EPB). This is the default value.
EPB	Return an encrypted PIN block and a MAC to verify the encrypted PIN block.
Script selection algorithm and method keyword (One, optional)	
AES-CBC	Specifies to use CBC mode to AES encrypt the script. If SCR2020 is specified, AES-CBC specifies to AES encrypt the PIN block plus additional data in the script.
NOSCRIPT	Do not return an encrypted SMPIN message with a MAC. This is the default value.
TDES-CBC	Specifies to use CBC mode to TDES encrypt the script. If SCR2020 is specified, TDES-CBC specifies to TDES CBC encrypt the PIN block plus additional data in the script.
TDES-ECB	Specifies to use ECB mode to TDES encrypt the script. If SCR2020 is specified, TDES-ECB specifies to TDES ECB encrypt the PIN block plus additional data in the script.
Script Process (One, optional). May be specified with keyword AES-CBC, TDES-CBC, or TDES-ECB. Otherwise, invalid.	
SCR2013	Specifies to use script processing rules introduced with the service in 2013. This is the default.
SCR2020	Specifies to use the script processing rules introduced with this service in 2020. The new process encrypts only the new PIN block and any additional data in the card_script_data field

	parameter and returns only the encrypted portion of the card_script_data field in the output script parameter. This keyword is introduced in APAR OA58880 for ICSF FMID HCR77D1.
Pin encryption keyword (One, optional) Only valid if AES-CBC, TDES-CBC or TDES-ECB is selected above.	
CLEARPIN	Do not encrypt the PIN prior to inserting in the script block. This is the default value.
SELF-ENC	Copy the PIN-block self-encrypted to the clear PIN block within the clear output message. Use this rule array keyword to specify that the 8-byte PIN block shall be used as a DES key to encrypt the PIN block. The service copies the self-encrypted PIN block to the clear PIN block in the output message.
MAC Ciphering Method (One required for AES-CBC, one optional for TDES-CBC or TDES-ECB; otherwise, not allowed.)	
CMAC	Specifies to use the cipher-based MAC algorithm block cipher mode of operation for authentication, recommended in NIST SP 800-38B. Required for AES-CBC. Only valid with AES-CBC.
EMVMACD	Specifies the EMV-related message-padding and calculation method. Not valid with AES-CBC.
TDES-MAC	Specifies the ANS X9.9 Option 1 (binary data) procedure and a CBC Triple-DES encryption of the data. Not valid with AES-CBC.
X9.19OPT	Specifies the ANS X9.19 Optional Procedure. A double-length key is required. Not valid with AES-CBC. This is the default value for TDES-CBC and TDES-ECB.
MAC Length and presentation (One, optional) Only valid if AES-CBC, TDES-CBC, or TDES-ECB is selected above.	
MACLEN8	Specifies a 8-byte MAC. This is the default value for TDES-CBC and TDES-ECB.
MACLEN16	Specifies a 16-byte MAC. Only valid with CMAC. This is the default for AES-CBC.
PIN Block format (One, optional)	
ISO-1	Specifies that the encrypted PIN block in the ISO_encrypted_PIN_block parameter is in the ISO-1 format. This is the default.
ISO-4	Specifies that the encrypted PIN block in the ISO_encrypted_PIN_block parameter is in the ISO-4 format.

PAN_data_length

Direction	**Type**
Input	Integer

Specifies the length in bytes of the PAN_data parameter. The value must be between 10 and 19, inclusive.

PAN_data

Direction	**Type**
Input	Character

The PAN data which the PIN is associated. The full account number, including check digit, should be included. This parameter is character data.

card_p_data_length

Direction	Type
Input	Integer

Specifies the length in bytes of the card_p_data parameter. The value must be between 2 and 256, inclusive.

card_p_data

Direction	Type
Input	String

The time-invariant card data (CDp), determined by the card issuer, which is used to differentiate between multiple cards for one account.

card_t_data_length

Direction	Type
Input	Integer

Specifies the length in bytes of the card_t_data parameter. The value must be between 2 and 256, inclusive.

card_t_data

Direction	Type
Input	String

The time-sensitive card data, determined by the card issuer, which, together with the account number and the card_p_data, specifies an individual card.

cur_ISO_PIN_block_length

Direction	Type
Input	Integer

The cur_ISO_pin_block_length specifies the length in bytes of the cur_ISO_PIN_block parameter. This value must be 8 when ISO-1 is specified and 16 when ISO-4 is specified.

cur_ISO_PIN_block

Direction	Type
Input	String

The encrypted PIN block with the current PIN in ISO-1 or ISO-4 format.

new_ISO_PIN_block_length

Direction	Type
Input	Integer

The new_ISO_pin_block_length specifies the length in bytes of the new_ISO_PIN_block parameter. This value must be 8 when ISO-1 is specified and 16 when ISO-4 is specified.

new_ISO_PIN_block

Direction	Type
Input	String

The new encrypted PIN block with the customer chosen PIN. The PIN block must be in ISO-1 or ISO-4 format.

card_script_data_length

Direction	Type
Input	Integer

The number of bytes of data in the card_script_data parameter. If the script selection of the rule array specifies to not return an encrypted SMPIN message with a PIN block MAC (that is, AES-CBC, TDES-CBC, or TDES- ECB is not specified), the value must be 0. When the script selection keyword is TDES-CBC or TDES-ECB, the value is a positive value less than or equal to 4096 and a multiple of 8. When the keyword is AES-CBC, the value is a positive value less than or equal to 4096.

card_script_data

Direction	Type
Input	String

The cleartext data to be MAC'd. The script_offset value can be considered the PIN block offset. If the SCR2013 keyword or no script process rule is specified, the entire field is encrypted and returned in the script parameter. If the SCR2020 keyword is specified, script_length bytes are encrypted starting at the offset indicated by the script_offset parameter. The PIN block plus additional data are encrypted and inserted at the offset specified by the script_offset parameter the MAC operation is performed. The smaller encrypted result is returned in the script parameter.

script_offset

Direction	Type
Input	Integer

The offset in bytes from the beginning of the cleartext data in the card_script_data variable to the location for the clear PIN block. The first byte of the cleartext data is offset 0. If the SCR2013 keyword or no script process rule is specified, this offset plus the script_offset_field_length must be less than or equal to the card_script_data_length. If the SCR2020 keyword is specified, the value of the script_offset plus the script_length must be less than or equal to the card_script_data_length.

script_offset_field_length

Direction	Type
Input	Integer

The number of bytes of data in the PIN block format referenced by the output PIN profile. Currently, this length must be 8. The PIN block must fit entirely within the card_script_data parameter. If NOSCRIPT is specified in the rule array, this parameter is ignored.

script_initialization_vector_length

Direction	Type
Input	Integer

Specifies the length in bytes of the script_initialization_vector parameter. For script selection algorithm and method keyword AES-CBC, the value must be 16. For TDES-CBC, the value must be 8. Otherwise, the value must be 0.

script_initialization_vector

Direction	Type
Input	String

The 8-byte or 16 byte initialization data for encrypting the script. The value of this parameter must be a string of hexadecimal zeroes. If the script_initialization_vector_length is 0, this parameter is ignored.

output_PIN_profile

Direction	Type
Input	String

A 24-byte string containing the PIN profile, including the PIN block format for the script. See “The PIN profile” on page 626 for additional information. You can use PIN-block formats ISO-0, ISO-1, ISO-2, ISO-3, and ISO-4 with this service. If NOSCRIPT is specified in the rule array, this parameter is ignored.

PIN_reference_value_length

Direction	Type
Input	Integer

Specifies the length in bytes of the PIN_reference_value parameter. This value must be 16.

PIN_reference_value

Direction	Type
Input	String

The 16-byte PIN reference value of the current PIN for comparison to the calculated value.

PRW_random_number_length

Direction	Type
Input	Integer

Specifies the length in bytes of the PRW_random_number parameter. The value must be 4.

PRW_random_number

Direction	Type
Input	String

The 4-byte random number associated with the PIN reference value of the current PIN.

PRW_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the PRW_key_identifier parameter. If the PRW_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

PRW_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to verify the PRW of the current PIN block. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be AES, the key type must be PINPRW, and the key usage fields must indicate VERIFY, CMAC, and DKPINOP.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

cur_IPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the cur_IPIN_encryption_key_identifier parameter. If the cur_IPIN_encryption_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

cur_IPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the PIN_block containing the current PIN. The key identifier is an operational token or the key label of an operational token in key storage.

When ISO-1 is specified, the key algorithm of this key must be DES and the key type must be IPINENC. The control vector must enable the verification of an encrypted PIN (EPINVER bit 19 = B'1').

When ISO-4 is specified, the key algorithm of this key must be AES and the key type must be PINPROT. The key usage fields must specify DECRYPT, CBC, NOFLDFMT, ISO-4, and PINXLATE. To use the same PINPROT key for the new and current inbound IPIN encryption key, the key usage fields must specify DECRYPT, CBC, NOFLDFMT, ISO-4, PINXLATE, and EPINVER.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

new_IPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the new_IPIN_encryption_key_identifier parameter. If the new_IPIN_encryption_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

new_IPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the PIN_block containing the new PIN. The key identifier is an operational token or the key label of an operational token in key storage.

When ISO-1 is specified, the key algorithm of this key must be DES and the key type must be IPINENC. The control vector must enable for translation (TRANSLAT bit 22 = B'1'). To use the same key token for the current and the new inbound PIN encryption key, the control vector must have key usages TRANSLAT and EPINVER enabled.

When ISO-4 is specified, the key algorithm of this key must be AES and the key type must be PINPROT. The key usage fields must specify DECRYPT, CBC, and PINXLATE. To use the same key token for the current and the new inbound PIN encryption key, the key usage EPINVER, and PINXLATE must be enabled.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

script_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the script_key_identifier parameter. If the rule array indicates that no script is to be processed, this value must be 0. If the script_key_identifier contains a label, the lengthmust be 64. Otherwise, the value must be between the actual length of the token and 725.

script_key_identifier

Direction	Type
Input/Output	String

The identifier of the key for encryption of the script. The key identifier is an operational token or the key label of an operational token in key storage. For script selection algorithm and method keyword AES-CBC, the key algorithm of the key must be AES, the key type must be SECMSG, and the key usage fields must indicate SMPIN and allow use by the CSNBDPC service (ANY-USE or DPC-ONLY). For keywords TDES-CBC or TDES-ECB, the key algorithm of this key must be DES, the key type must be SECMSG with the SMPIN usage bit (CV bit 19) set to B'1'.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

script_MAC_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the script_MAC_key_identifier parameter. If the rule array indicates that no script is to be processed, this value must be 0. If the script_MAC_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

script_MAC_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to generate the MAC of the script. The key identifier is an operational token or the key label of an operational token in key storage. For script selection algorithm and method keyword AES-CBC, the key algorithm of the key must be AES, the key type must be MAC, and the key usage fields must indicate GENERATE or GENONLY and CMAC. For keywords TDES-CBC or TDES-ECB, the key algorithm of this key must be DES, the key type must be MAC, and the key must be double-length.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

new_PRW_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the new_PRW_key_identifier parameter. If the new_PRW_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

new_PRW_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to verify the new PRW. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be AES, the key type must be PINPRW, and the key usage fields must indicate GENONLY, CMAC, and DKPINOP.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

OPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the OPIN_encryption_key_identifier parameter. If the rule array indicates that no encrypted PIN block is to be returned, this value must be 0. If the OPIN_encryption_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

OPIN_encryption_key_identifier

Direction	Type

Input/Output	String

The identifier of the key to encrypt the new PIN block. The key identifier is an operational token or the key label of an operational token in key storage. If the OPIN_encryption_key_identifier_length is 0, this parameter is ignored. The key algorithm of this key must be AES, the key type must be PINPROT, and the key usage fields must indicate ENCRYPT, CBC, and DKPINOP.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

OEPB_MAC_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the OEPB_MAC_key_identifier parameter. If the rule array indicates that no encrypted PIN block MAC is to be returned, this value must be 0. If the OEPB_MAC_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

OEPB_MAC_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to generate the MAC of new PIN block. The key identifier is an operational token or the key label of an operational token in key storage. If the OEPB_MAC_key_identifier_length is 0, this parameter is ignored. The key algorithm of this key must be AES, the key type must be MAC, and the key usage fields must indicate CMAC, GENONLY, and DKPINOP.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

script_length

Direction	Type
Input/Output	Integer

The number of bytes of data in the script variable. The value must be 0 if the script selection algorithm and method of the rule array specifies NOSCRIPT. For scripting, if the SCR2020 keyword is specified, the value must be set to the PIN block length plus the length of the additional customer defined data. Otherwise, the value of the script_offset plus the script_length must be less than or equal to the card_script_data_length.

script

Direction	Type
Output	String

The script returned. If the rule array specifies to return a script, script_length bytes of this variable are overwritten. If SCR2020 is specified, the parameter contains the encrypted part of the script. Otherwise, it contains the entire script.

script_MAC_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the script_MAC parameter. If the NOSCRIPT keyword is selected, this value must be 0. When the MAC length keyword is MACLEN8 or MACLEN16, the value must be at least as large as indicated by the keyword specified. When no MAC length keyword is specified and the script selection keyword is AES-CBC, the value must be between 4 and 16 inclusive. On output, the parameter is updated with the actual length of the script_MAC parameter.

script_MAC

Direction	Type
Output	String

The 8 byte or 16 byte MAC of the encrypted script. If the script_MAC_length is 0, this parameter is ignored.

new_PIN_reference_value_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the new_PIN_reference_value parameter. The value must be at least 16. On output, it will be set to 16.

new_PIN_reference_value

Direction	Type
Output	String

The 16-byte new PIN reference value of the new PIN block.

new_PRW_random_number_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the new_PRW_random_number parameter. The value must be at least 4. On output, it will be set to 4.

new_PRW_random_number

Direction	Type
Output	String

The 4-byte random number associated with the new PIN reference value.

output_encrypted_PIN_block_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the output_encrypted_PIN_block parameter. If the rule array indicates that no encrypted PIN block should be returned, this value must be 0. Otherwise, it should be at least 32. On output it will be set to 32.

output_encrypted_PIN_block

Direction	Type
Output	String

The 32-byte encrypted new PIN block. If the output_encrypted_PIN_block_length is 0, this parameter is ignored.

PIN_block_MAC_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the PIN_block_MAC parameter. If the rule_array indicates that no PIN block MAC should be returned, this value must be 0. Otherwise, it must be at least 8.

PIN_block_MAC

Direction	Type
Output	String

The 8-byte MAC of the new encrypted PIN block. If the PIN_block_MAC_length is 0, this parameter is ignored.

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS.

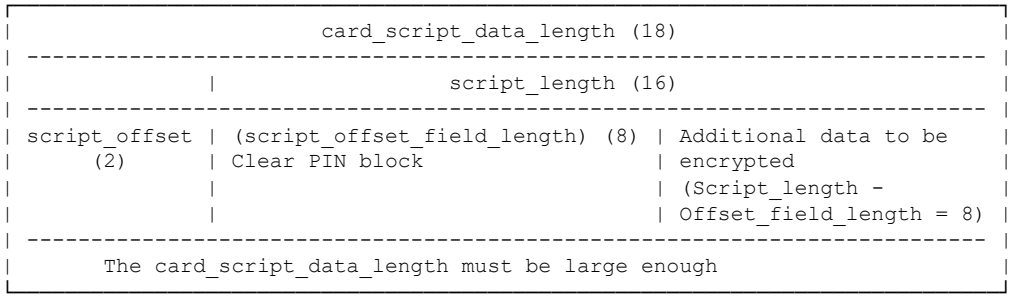
Notes on script processing

When using the SCR2020 keyword, the script, whose length is represented by script_length, is now only a part of the card_script_data area, whose length is represented by card_script_data_length. This script length will indicate how much of the card_script_data area is to be encrypted. Therefore, the script_offset + the script_length must not be less than the card_script_data_length. If it is, a size error is returned.

Note: If the script_offset = 0, the card_script_data_length and the script_length could be equal.

Here is an example showing the correct layout with sample input lengths:

- card_script_data_length = 18
- script_offset = 2
- script_offset_field_length = 8
- script_length = 16



Note: It is not possible with SCR2020 to encrypt data before the clear PIN block. This is a difference from SCR2013. SCR2020 is mandatory for ISO-4 PIN blocks. Therefore, this restriction applies to all scripts that handle ISO-4 PINs.

Access control points

The **DK PIN Change** access control point in the domain role controls the function of this service.

When the **General ISO PIN Error Mode** access control is enabled, the return code will be a general PIN block error (8/2514) instead of some of the PIN block errors return code. The use of a general return code can prevent the abuse of PIN processing error messages due to information leakage derived from the return code reason codes returned under various conditions. See 'PIN block error processing mode' on page 80 for details.

When the **Disallow PIN block format ISO-1** access control is enabled in the domain role, the PIN block format rule_array keyword ISO-1 is not allowed.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in "CCA release levels," on page 173.

Table 63. DK PIN Change required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC		This service is not supported.
IBM System z10 EC IBM System z10 BC		This service is not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code (LIC). Rule array keywords AES-CBC, CMAC, ISO-1, and ISO-4 are not supported. Triple-length DES keys are not supported. Compliant-tagged key tokens are not supported. Keywords SCR2013 and SCR2020 are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code (LIC). Rule array keywords AES-CBC, CMAC, and MACLEN16 require the June 2015 or later licensed internal code (LIC). Keywords ISO-1 and ISO-4 are not supported.

		Triple-length DES keys are not supported. Compliant-tagged key tokens are not supported. Keywords SCR2013 and SCR2020 are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Rule array keywords AES-CBC, CMAC, and MACLEN16 require the July 2015 or later licensed internal code (LIC). Keywords ISO-1 and ISO-4 and triple-length DES keys require the July 2019 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. The combination of keyword ISO-4 and any keyword form the script selection algorithm and method keyword group requires the June 2020 or later licensed internal code (LIC). Keywords SCR2013 and SCR2020 require the June 2020 or later licensed internal code (LIC).
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Rule array keywords AES-CBC, CMAC, and MACLEN16 require the July 2015 or later licensed internal code (LIC). Keywords ISO-1 and ISO-4 and triple-length DES keys require the December 2018 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported. The combination of keyword ISO-4 and any keyword form the script selection algorithm and method keyword group requires the June 2020 or later licensed internal code (LIC). Keywords SCR2013 and SCR2020 require the June 2020 or later licensed internal code (LIC).
	Crypto Express6 CCA Coprocessor	Keywords ISO-1 and ISO-4 and triple-length DES keys require the December 2018 or later licensed internal code (LIC).

		Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC). The combination of keyword ISO-4 and any keyword form the script selection algorithm and method keyword group requires the June 2020 or later licensed internal code (LIC). Keywords SCR2013 and SCR2020 require the June 2020 or later licensed internal code (LIC).
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported. The combination of keyword ISO-4 and any keyword form the script selection algorithm and method keyword group requires the June 2020 or later licensed internal code (LIC). Keywords SCR2013 and SCR2020 require the June 2020 or later licensed internal code (LIC).
	Crypto Express6 CCA Coprocessor Crypto Express7 CCA Coprocessor	The combination of keyword ISO-4 and any keyword form the script selection algorithm and method keyword group requires the June 2020 or later licensed internal code (LIC). Keywords SCR2013 and SCR2020 require the June 2020 or later licensed internal code (LIC).

DK PIN Verify (CSNBDPV and CSNEDPV)

Use the DK PIN Verify callable service to verify an ISO-1 or ISO-4 format PIN. The input PIN will be converted to PBF-0 format. A test PIN reference value (PRW) is created and that value is bitwise compared to the input PRW.

The callable service name for AMODE(64) invocation is CSNEDPV.

Format

```
CALL CSNBDPV(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              PAN_data_length,
```

```
PAN_data,
card_data_length,
card_data,
PIN_reference_value_length,
PIN_reference_value,
PRW_random_number_length,
PRW_random_number,
ISO_encrypted_PIN_block_length,
ISO_encrypted_PIN_block,
PRW_key_identifier_length,
PRW_key_identifier,
IPIN_encryption_key_identifier_length,
IPIN_encryption_key_identifier)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. The value can be 0 or 1.

rule_array

Direction	Type
Input	Character

Keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 64. Keywords for the DK PIN Verify Service

Keyword	Meaning
PIN Block format (One, optional)	
ISO-1	Specifies that the encrypted PIN block in the ISO_encrypted_PIN_block parameter is in the ISO-1 format. This is the default.
ISO-4	Specifies that the encrypted PIN block in the ISO_encrypted_PIN_block parameter is in the ISO-4 format.

PAN_data_length

Direction	Type
Input	Integer

Specifies the length in bytes of the PAN_data parameter. The value must be between 10 and 19, inclusive.

PAN_data

Direction	Type
Input	Character

The PAN data which the PIN is associated. The full account number, including check digit, should be included. This parameter is character data.

card_data_length

Direction	Type
Input	Integer

Specifies the length in bytes of the card_data parameter. The value must be between 4 and 512, inclusive.

card_data

Direction	Type
Input	String

The time-invariant card data (CDp) and the time-sensitive card data (CDt) which, together with the account number, specifies an individual card.

PIN_reference_value_length

Direction	Type

Input	Integer

Specifies the length in bytes of the PIN_reference_value parameter. This value must be 16.

PIN_reference_value

Direction	Type
Input	String

The 16-byte PIN reference value for comparison to the calculated value.

PRW_random_number_length

Direction	Type
Input	Integer

Specifies the length in bytes of the PRW_random_number parameter. The value must be 4.

PRW_random_number

Direction	Type
Input	String

The 4-byte random number associated with the PIN reference value.

ISO_encrypted_PIN_block_length

Direction	Type
Input	Integer

Specifies the length in bytes of the ISO_encrypted_PIN_block parameter. This value must be 8 for a ISO-1 block and 16 for a ISO-4 block.

ISO_encrypted_PIN_block

Direction	Type
Input	String

The 8-byte encrypted PIN block in ISO-1 format.

PRW_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the PRW_key_identifier parameter. If the PRW_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

PRW_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to verify the PIN reference value. The key identifier is an operational token or the key label of an operational token in key storage. The key

algorithm of this key must be AES, the key type must be PINPRW, and the key usage fields must indicate VERIFY, CMAC, and DKPINOP.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

IPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the IPIN_encryption_key_identifier parameter. If the IPIN_encryption_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 725.

IPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the PIN_block. The key identifier is an operational token or the key label of an operational token in key storage.

When ISO-1 is specified, the key algorithm of this key must be DES and the key type must be IPINENC. The control vector must enable the verification of an encrypted PIN (EPINVER bit 19 = B'1').

When ISO-4 is specified, the key algorithm of this key must be AES and the key type must be PINPROT. The key usage fields must specify DECRYPT, CBC, NOFLDFMT, and EPINVER.

If the token supplied was encrypted under the old master key, the token will be returned encrypted under the current master key.

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS.

Access control points

The DK PIN Verify access control point in the domain role controls the function of this service.

When the **General ISO PIN Error Mode** access control is enabled, the return code will be a general PIN block error (8/2514) instead of some of the PIN block errors return code. The use of a general return code can prevent the abuse of PIN processing error messages due to information leakage derived from the return code reason codes returned under various conditions. See ‘PIN block error processing mode’ on page 80 for details.

When the Disallow PIN block format ISO-1 access control is enabled in the domain role, the PIN block format rule_array keyword ISO-1 is not allowed.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 65. DK PIN Verify required hardware		
Server	**Required cryptographic hardware**	**Restrictions**
IBM System z9 EC IBM System z9 BC		This service is not supported.
IBM System z10 EC IBM System z10 BC		This service is not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code (LIC). Keywords ISO-1 and ISO-4 are not supported. Triple-length DES keys are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code (LIC). Keywords ISO-1 and ISO-4 are not supported. Triple-length DES keys are not supported. Compliant-tagged key tokens are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Keywords ISO-1 and ISO-4 and triple-length DES keys require the July 2019 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Keywords ISO-1 and ISO-4 and triple-length DES keys require the December 2018 or later licensed internal code (LIC). Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Keywords ISO-1 and ISO-4 and triple-length DES keys require the December 2018 or later licensed internal code (LIC). Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC).
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Compliant-tagged key tokens are not supported.

	Crypto Express6 CCA Coprocessor Crypto Express7 CCA Coprocessor	

Using digital signatures

Signature algorithms and formatting methods

ICSF supports signature generation and verification for RSA, EC, and CRYSTALS-Dilithium algorithms. This topic lists the hashing algorithms supported by CSNDDSG and CSNDDSV that are either recommended or required by the standard for the algorithms and formatting methods. ICSF will only enforce the hashing algorithm when required by a standard.

ICSF supports these hash formatting methods for the RSA algorithm:

ANSI X9.31

Required hash methods: RIPEMD-160, SHA-1, SHA-256, SHA-384, or SHA-512.

ISO 9796-1

Recommended hash methods: Any.

RSA PKCS 1.0 and PKCS 1.1

Recommended hash methods: MD5, SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512.

RSA PKCS-PSS

Required hash methods: SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512.

Padding on the left with zeros

Recommended hash methods: Any.

ICSF supports these elliptic curve algorithms:

ANSI X9.62 ECDSA (Brainpool, NIST prime, and Koblitz)

Recommended hash methods: SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512.

ISO/IEC 14888 Schnorr Digital Signature Algorithm (EC-SDSA)

- The hash method for P256 keys is SHA-256.
- The hash method for P521 keys is SHA-512.

Edwards-curve Digital Signature Algorithm (EdDSA)

- Required hash methods for Ed25519: SHA-512.
- Required hash methods for Ed448: SHAKE-256.

ICSF supports these hash formatting methods for the CRYSTALS-Dilithium algorithm:

CRYSTALS-Dilithium Digital Signature Algorithm (CRDL-DSA)

Required hash methods for CRYSTALS-Dilithium: SHAKE-256.

Digital Signature Generate (CSNDDSG and CSNFDSG)

Use the Digital Signature Generate callable service to generate a digital signature using a PKA private key or, for some limited functions, a secure PKCS #11 private key. Private keys must be valid for signature usage.

This service supports these hash formatting methods for the RSA algorithm:

- ANSI X9.31
- ISO 9796-1
- RSA DSI PKCS #1 v1.5 and v2.1
- Padding on the left with zeros

This service supports the ANSI X9.62 ECDSA (Brainpool, NIST prime, and Koblitz), the Edwards-curve Digital Signature Algorithm (EdDSA), the ISO/IEC 14888 Schnorr Digital Signature Algorithm (EC-SDSA), and the CRYSTALS-Dilithium Digital Signature Algorithm (CRDL-DSA) algorithms.

This service accepts either the input message or a hash of the input message.

For CCA keys, when the private_key_identifier parameter specifies:

An RSA private key

Select the method of formatting the text by using the digital signature formatting method rule array keyword.

An ECC private key

Select the ECDSA, EDDSA, or EC-SDSA signature algorithm rule array keyword.

For the EC-SDSA algorithm, the key is restricted to secp256r1 (P256) and secp521r1 (P521) curves. The hash method for P256 keys is SHA-256. The hash method for P521 keys is SHA-512.

A CRYSTALS-Dilithium private key

Select the CRDL-DSA signature algorithm rule array keyword.

For secure PKCS #11 keys, when the private_key_identifier parameter specifies:

An RSA private key

Select the PKCS-1.1 formatting method keyword.

An ECC private key

Select the ECDSA or EDDSA algorithm keyword.

If keyword ECDSA is specified in the rule array, the Elliptic Curve Digital Signature Algorithm is used as the digital-signature hash formatting method. If keyword EDDSA is specified, the EdDSA algorithm and hashing method appropriate for Edwards curves is used. Otherwise, specify the optional digital-signature hash formatting method keyword in the rule array for the method used to generate the RSA digital signature.

The callable service name for AMODE(64) invocation is CSNFDSG.

Format

```
CALL CSNDDSG(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              private_key_identifier_length,
              private_key_identifier,
              data_length,
              data,
              signature_field_length,
              signature_bit_length,
              signature_field)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program.Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you are supplying in the rule_array parameter. The value may be 0 1, 2, 3, or 4.

rule_array

Direction	Type
Input	String

The rule_array contains keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 66. Keywords for Digital Signature Generate Control Information

Keyword	Meaning
Signature algorithm (One, optional)	
CRDL-DSA	Specifies to generate a signature in the CRYSTALS-Dilithium digital signature scheme.
ECDSA	Specifies to generate a digital signature using the ECDSA algorithm.
EC-SDSA	Specifies to generate a digital signature using the EC-SDSA algorithm.
EDDSA	Specifies to generate a digital signature using the EdDSA algorithm.
RSA	Specifies to generate an RSA digital signature. This is the default.
Digital Signature Formatting Method (optional, valid for RSA digital signature generation only)	
ISO-9796	Specifies to format the hash according to the ISO/IEC 9796-1 standard and generate the digital signature. Any hash method is allowed. This is the default.
PKCS-1.0	Specifies to format the digital signature on the string supplied in the hash variable as defined in the RSA PKCS #1 standard for block-type 00. The PKCS #1 v2.0 standard no longer defines this signature scheme.
PKCS-1.1	Specifies to format the digital signature on the string supplied in the hash variable as defined in the RSA PKCS #1 v2.0 standard for the RSASSA-PKCS1-v1_5 signature scheme. This was formerly known as block-type 01.
PKCS-PSS	Specifies to format the hash as defined in the RSA PKCS #1 v2.2 standard for the RSASSA-PSS signature scheme. Only valid for RSA-AESM and RSA-AESC key tokens.
X9.31	Specifies to format the hash according to the ANSI X9.31 standard. The modulus length of a key used must be one of 1024, 1280, 1536, 1792, 2048, or 4096 bits.
ZERO-PAD	Specifies to format the hash by padding it on the left with binary zeros to the length of the RSA key modulus. Any supported hash function is allowed.
Data Input Type (One, optional)	
HASH	Specifies that the data parameter contains the hash that is to be signed. This is the default.
MESSAGE	Specifies that the data parameter contains the message that is to be hashed and signed. This keyword is required with EDDSA, EC-SDSA, and CRDL-DSA keywords.
Hash Method Specification (One required:) *• When the MESSAGE keyword is specified,*	

• When the HASH keyword is specified and the hash formatting method is PKCS-PSS or X9.31, or *• When the signature algorithm keyword EDDSA or CRDL-DSA is specified.*	
ED-HASH	Process the text supplied in the data variable using the hash method for the Edwards curve of the key passed in the PKA_private_key_identifier parameter. For Ed25519, this is SHA-512. For Ed448, this is SHAKE-256. This keyword is required with the EDDSA keyword.
CRDLHASH	Process the text supplied in the message variable using the hash method appropriate for the CRYSTALS-Dilithium algorithm. This is SHAKE-256. Required for CRDL-DSA.
MD5	Process the text supplied in the data parameter using the MD5 hash method. Not valid with PKCS-PSS or X9.31.
RPMD-160	Process the text supplied in the data parameter using the RIPEMD-160 hash method. Not valid with PKCS-PSS.
SHA-1	Process the text supplied in the data parameter using the SHA-1 hash method.
SHA-224	Process the text supplied in the data parameter using the SHA-224 hash method. Not valid with X9.31.
SHA-256	Process the text supplied in the data parameter using the SHA-256 hash method.
SHA-384	Process the text supplied in the data parameter using the SHA-384 hash method.
SHA-512	Process the text supplied in the data parameter using the SHA-512 hash method.

private_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the private_key_identifier parameter. If the private_key_identifier contains a label, the value must be 64. Otherwise, the value must be between the actual length of the token and 6500.

For secure PKCS #11 keys, the length must be 64.

private_key_identifier

Direction	Type
Input	String

The identifier of the private key to generate the signature. The key identifier is an RSA, EC, or CRYSTALS-Dilithium private key token, or label of such a private key identifier in key storage.

For CCA, this is an RSA, EC, or CRYSTALS-Dilithium private key or a retained RSA key. The token key usage flag must permit the generation of signatures. The format restriction field of the RSA private key token will restrict the format allowed to be used with the key.

If formatting method PKCS-PSS is specified, the RSA key token must have an AES object protection key (private key section identifiers X'30' and X'31'), which can be created with key type RSA-AESM and RSA-AESC in PKA Key Token Build (CSNDPKB and CSNFPKB).

For secure PKCS #11 keys, this is the 44-byte handle of the private key, prefixed with an EBCDIC equal sign character (‘=’ or X’7E’), and padded on the right with spaces for a total length of 64 bytes.

data_length

Direction	Type
Input	Integer

The length of the data parameter in bytes. See Usage Notes for details of the formatting methods and the length requirements.

When the data input type keyword is MESSAGE, the value is the length of the message to be hashed. The maximum value is 2147483647 bytes. When the signature algorithm keyword is EDDSA or EC-SDSA, the maximum value is 8192. When CRDL-DSA is specified, the maximum value is 5000. A hash method rule array keyword must be specified.

When the data input type keyword is HASH, the value must be the exact length of the hash to be signed. The maximum value is 516 bytes. The length of the hash for the supported hashing method is 16 for MD5, 20 for SHA-1 and RPMD-160, 28 for SHA-224, 32 for SHA-256, 48 for SHA-384, and 64 for SHA-512. You can use either the One-Way Hash Generate callable service or the MDC Generate callable service to generate the hash.

For the PKCS-PSS formatting rule, the first four bytes must be the salt length. The remaining bytes of the parameter must contain the hash or message. The value will be 4 + length of the hash or message.

For the ZERO-PAD format rule, the length is restricted to 36 for RSA keys that permit key management in the key usage field. For RSA keys that permit signature only in the key usage field, the maximum value is 512. This hash length limit is controlled by an access control point. Only RSA keys that permit key management are affected by this access control point. See the restrictions section of this service for details.

data

Direction	Type
Input	String

The application-supplied data to be signed. The data can be the message to be hashed and signed or the hash of the message. See Usage Notes for details of the formatting methods and the data requirements.

The data must be in the caller's primary address space.

For the PKCS-PSS formatting rule, the first four bytes must be the salt length. The remaining bytes of the parameter must contain the hash or message.

signature_field_length

Direction	Type
Input/Output	Integer

The length in bytes of the signature_field to contain the generated digital signature. Upon return, this field contains the actual length of the generated signature. The maximum size

for RSA is 512. The maximum size of ECC is 132. For CRYSTALS-Dilithium, the maximum size is 3500.

For RSA, this must be at least the byte length of the modulus rounded up to a multiple of 32 bytes for the X9.31 signature format or one byte for all other signature formats.

For ECDSA NIST prime curves, the maximum is 2 * 521 bits. For brain pool curves, the maximum size is 2 * 512 bits. For Koblitz secp256k1 curves, the maximum is 64 bytes.

signature_bit_length

Direction	Type
Output	Integer

The bit length of the digital signature generated. For ISO-9796, this is 1 less than the modulus length. For other RSA processing methods, this is the modulus length.

signature_field

Direction	Type
Input/Output	String

The digital signature generated is returned in this field. The digital signature is in the low-order bits (right-justified) of a string whose length is the minimum number of bytes that can contain the digital signature. This string is left-justified within the signature_field. Any unused bytes to the right are undefined.

Restrictions

Although ISO-9796 does not require the input hash to be an integral number of bytes in length, this service requires you to specify the hash_length in bytes.

For CCA RSA keys, the hash length limit is controlled by the DSG ZERO-PAD unrestricted hash length access control point. If enabled, the maximum hash length limit for ZERO-PAD is the modulus length of the PKA private key. If disabled, the maximum hash length limit for ZERO-PAD is 36 bytes. Only RSA key management keys are affected by this access control point. The limit for RSA signature use only keys is 512 bytes. This access control point is disabled in the domain role. You must have a TKE workstation to enable it.

Authorization

To use this service with a secure PKCS #11 private key that is a public object, the caller must have SO (READ) authority or USER (READ) authority (any access) to the containing PKCS #11 token.

To use this service with a secure PKCS #11 private key that is a private object, the caller must have USER (READ) authority (user access) to the containing PKCS #11 token.

See z/OS Cryptographic Services ICSF Writing PKCS #11 Applications for more information on the SO and User PKCS #11 roles.

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS, PKDS, or TKDS.

Notes on formatting the message:

ISO-9796

The length of the hash must be less than or equal to one-half of the number of bytes required to contain the modulus of the RSA key.

PKCS-1.0 and PKCS-1.1

The length of the hash must be 11 bytes shorter than the number of bytes required to contain the modulus of the RSA key.

When the HASH keyword is specified, the hash must be BER encoded. See “Formatting hashes and keys in public-key cryptography” on page 1472 for a description of the formatting methods.

PKCS-PSS

The first four bytes of the data parameter will contain the salt length. The remaining bytes of the parameter contains the hash or message.

It is recommended that the salt length be either 0 or the byte length of the hash algorithm selected. The salt length should not be less than the byte length of the hash algorithm, but it can be greater. When the Digital Signature Generate – PKCS-PSS allow small salt access control is enabled in the domain role, the salt length may be less than the length of the hash.

The size of the data to be signed is governed by the size of the RSA modulus. The modulus size and hash length affect the maximum salt length for a given key modulus size and specified hash. The maximum salt length equals modulus size/8 - hash length - 2. For example, with a 4096 bit modulus key and SHA-1, the maximum salt length becomes – (4096/8) – 20 -2 => 512-20-2 = 490.

X9.31

There are no restrictions for the hash length or message.

ZERO-PAD

The length of the hash must be less than or equal to the number of bytes required to contain the modulus of the RSA key.

ECDSA

There are no restrictions for the hash length or message.

EdDSA and EC-SDSA

The length of the message must be less than or equal to 8192 bytes.

CRDL-DSA

The length of the message must be less than or equal to 5000 bytes.

The Digital Signature Generate callable service can take advantage of a PKCS#11 private key object stored in the TKDS, but you should check if using “PKCS #11 Private Key Sign (CSFPPKS and CSFPPKS6)” on page 1221 aligns better with the overall design of your application.

PKCS-PSS details

Before specifying PKCS-PSS, see section A.2.3 of RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2 (RFC 8017 (tools.ietf.org/html/rfc8017)). Section A.2.3 defines a parameter field for RSASSAPSS that has the following four parameters:

hashAlgorithm

This parameter identifies the hash function. A hashing-method specification keyword of SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512 is required. There is no default.

maskGenAlgorithm

This parameter identifies the mask generation function. MGF1 is a mask generation function based on a hash function and is the only function currently defined by the standard. For CCA, the hash function on which MGF1 is based is always the same as hashAlgorithm. Although this is not required, CCA enforces the recommendation by the standard that the underlying hash function be the same as the one identified by hashAlgorithm.

saltLength

This is the length of the salt, which is a randomly generated value. Use the data variable of the verb to prepend a 4-byte saltLength field in big endian format to the hash digest to be signed or the message to be hashed and signed.

trailerField

This is the trailer field number. This is not an input to the verb because it is always set to the value X'BC'. Other trailer field numbers are not supported by the standard.

Access control points

For PKA private keys, **the Digital Signature Generate** access control point controls the function of this service.

The length of the hash for ZERO-PAD is restricted to 36 bytes. If the **DSG ZERO-PAD unrestricted hash length** access control point is enabled in the domain role, the length of the hash is not restricted. This access control is disabled by default.

For the PKCS-PSS formatting method, ICSF requires the salt length specified in the data parameter to be zero or the length of the hash specified. When the **Digital Signature Generate – PKCS-PSS allow small salt** access control is enabled in the domain role, the salt length may be less than the length of the hash.

For secure PKCS #11 private keys, the Sign with private keys access control point controls the function of this service. For more information on the access control points of the Enterprise PKCS #11 coprocessor, see 'PKCS #11 Access Control Points' in z/OS Cryptographic Services ICSF Writing PKCS #11 Applications.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 67. Digital Signature Generate required hardware

Server	Required cryptographic hardware	Restrictions

IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	ECC not supported. CRYSTALS-Dilithium not supported. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Keywords PKCS-PSS, SHA-384, and SHA-512 are not supported. Compliant-tagged key tokens are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	ECC not supported. CRYSTALS-Dilithium not supported. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Keyword PKCS-PSS is not supported. The combination of the X9.31 keyword and either SHA-384 or SHA-512 is not supported. Compliant-tagged key tokens are not supported.
	Crypto Express3 Coprocessor	ECC support requires the Sep. 2010 licensed internal code (LIC). Keywords PKCS-PSS, EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. The combination of the X9.31 keyword and either SHA-384 or SHA-512 is not supported ECC Koblitz curve secp256k1 is not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	Keywords PKCS-PSS EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. Compliant-tagged key tokens are not supported
IBM zEnterprise EC12	Crypto Express3 Coprocessor	Keywords PKCS-PSS EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported.

IBM zEnterprise BC12	Crypto Express4 CCA Coprocessor	ECC Koblitz curve secp256k1 is not supported. Compliant-tagged key tokens are not supported.
	Crypto Express4 Enterprise PKCS #11 coprocessor	Required to use a secure PKCS #11 private key. Compliant-tagged key tokens are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	PKCS-PSS support requires the October 2016 or later licensed internal code (LIC). Keywords EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. Compliant-tagged key tokens are not supported.
	Crypto Express5 Enterprise PKCS #11 coprocessor	Required to use a secure PKCS #11 private key. Compliant-tagged key tokens are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	PKCS-PSS support requires the October 2016 or later licensed internal code (LIC). Keywords EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC). Keywords EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported.
	Crypto Express5 Enterprise PKCS #11 coprocessor Crypto Express6 Enterprise PKCS #11 coprocessor	Required to use a secure PKCS #11 private key. Compliant-tagged key tokens are not supported.

IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	PKCS-PSS support requires the October 2016 or later licensed internal code (LIC). Keywords EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Keywords EDDSA, EC-SDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported.
	Crypto Express7 CCA Coprocessor	Keywords EDDSA, ED-HASH, CRDL-DSA, and CRDLHASH require the June 2020 or later licensed internal code (LIC). ECC Koblitz curve secp256k1 requires the September 2020 or later licensed internal code (LIC). Keyword EC-SDSA requires the CCA release 7.4 or later licensed internal code.
	Crypto Express5 Enterprise PKCS #11 coprocessor Crypto Express6 Enterprise PKCS #11 coprocessor Crypto Express7 Enterprise PKCS #11 coprocessor	Required to use a secure PKCS #11 private key. Compliant-tagged key tokens are not supported.
	CP Assist for Cryptographic Functions	Clear and secure ECC key requests for the Prime P-256, Prime P-384, Prime P-521, Ed25519, and Ed448 curves are supported. Secure key support requires a CEX7 CCA coprocessor with the June 2020 or later licensed internal code.

Digital Signature Verify (CSNDDSV and CSNFDSV)

Use the Digital Signature Verify callable service to verify a digital signature using a PKA public key.

- The Digital Signature Verify callable service can use the RSA, EC, or CRYSTALS-Dilithium public key, depending on the digital signature algorithm used to generate the signature.
- The Digital Signature Verify callable service can use the RSA public keys that are contained in trusted blocks regardless of whether the block also contains rules to govern its use when

generating or exporting keys with the RKX service. If the TPK-ONLY keyword is used in the rule_array parameter, an error will occur if the PKA_public_key_identifier parameter does not contain a trusted block.
- The Digital Signature Verify callable service can use an X.509 certificate containing an RSA or ECC public key.

This service supports these hash formatting methods for RSA keys:

- ANSI X9.31
- ISO 9796
- RSA DSI PKCS #1 v2.0 and v2.2
- Padding on the left with zeros

This service supports the ANSI X9.62 ECDSA (Brainpool, NIST prime, and Koblitz), Edwards-curve Digital Signature Algorithm (EdDSA), the ISO/IEC 14888 Schnorr Digital Signature Algorithm (EC-SDSA), and CRYSTALS-Dilithium Digital Signature Algorithm (CRDL-DSA) algorithms.

This service accepts either the input message or a hash of the input message.

If keyword ECDSA is specified in the rule array, the Elliptic Curve Digital Signature Algorithm is used as the digital-signature hash formatting method. If keyword EDDSA is specified, the EdDSA algorithm and hashing method appropriate for Edwards curves is used. If keyword CRDL-DSA is specified, the CRYSTALS-Dilithium algorithm and hashing method appropriate for CRYSTALS-Dilithium is used. Otherwise, specify the optional digital-signature hash formatting method keyword in the rule array for the method used to generate the RSA digital signature being verified.

The callable service name for AMODE(64) invocation is CSNFDSV.

Format

```
CALL CSNDDSV(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              PKA_public_key_identifier_length,
              PKA_public_key_identifier,
              data_length,
              data,
              signature_field_length,
              signature_field)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you are supplying in the rule_array parameter. The value must be 0, 1, 2, 3, 4, 5, or 6.

rule_array

Direction	Type
Input	String

The rule_array contains keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 68. Keywords for Digital Signature Verify Control Information

Keyword	Meaning
Signature Algorithm (One, optional)	
CRDL-DSA	Specifies to verify a signature in the CRYSTALS-Dilithium digital signature scheme.
ECDSA	Specifies to verify a digital signature using the ECDSA algorithm.
EC-SDSA	Specifies to verify a digital signature using the EC-SDSA algorithm.
EDDSA	Specifies to verify a digital signature using the EDDSA algorithm.
RSA	Specifies to verify an RSA digital signature. This is the default.
Digital Signature Formatting Method (One, optional, RSA only)	
ISO-9796	Specifies to format the hash according to the ISO/IEC 9796-1 standard and generates the digital signature. Any hash method is allowed. This is the default.

PKCS-1.0	Specifies to format the digital signature on the string supplied in the hash variable as defined in the RSA PKCS #1 standard for block-type 00. The PKCS #1 v2.0 standard no longer defines this signature scheme.
PKCS-1.1	Specifies to format the digital signature on the string supplied in the hash variable as defined in the RSA PKCS #1 v2.0 standard for the RSASSAPKCS1-v1_5 signature scheme. This was formerly known as block-type 01.
PKCS-PSS	Specifies to format the hash as defined in the RSA PKCS #1 v2.2 standard for the RSASSA-PSS signature scheme.
X9.31	Specifies to format the hash according to the ANSI X9.31 standard. The input text must have been previously hashed with any of the hash algorithms listed. The modulus length of a key used must be one of 1024, 1280, 1536, 1792, 2048, or 4096 bits.
ZERO-PAD	Specifies to format the hash by padding it on the left with binary zeros to the length of the RSA key modulus. Any hash method is allowed.
Data Input Type (One, optional)	
HASH	Specifies that the data parameter contains the hash that is to be signed. This is the default.
MESSAGE	Specifies that the data parameter contains the message that is to be hashed and signed. This keyword is required with EDDSA, EC-SDSA, and CRDL-DSA keywords.
Hash Method Specification (One required:) *• When the MESSAGE keyword is specified,* *• When the HASH keyword is specified and the hash formatting method is PKCS-PSS or X9.31, or* *• When the signature algorithm keyword EDDSA or CRDL-DSA is specified.*	
ED-HASH	Process the text supplied in the data variable using the hash method for the Edwards curve of the key passed in the PKA_private_key_identifier parameter. For Ed25519, this is SHA-512. For Ed448, this is SHAKE-256. This keyword is required with the EDDSA keyword.
CRDLHASH	Process the text supplied in the message variable using the hash method appropriate for the CRYSTALS-Dilithium algorithm. This is SHAKE-256. Required for CRDL-DSA.
MD5	Process the text supplied in the data parameter using the MD5 hash method. Not valid with PKCS-PSS.
RPMD-160	Process the text supplied in the data parameter using the RIPEMD-160 hash method. Not valid with PKCS-PSS.
SHA-1	The input value supplied in the data parameter is generated using the SHA-1 hash method.
SHA-224	The input value supplied in the data parameter is generated using the SHA-224 hash method.
SHA-256	The input value supplied in the data parameter is generated using the SHA-256 hash method.
SHA-384	The input value supplied in the data parameter is generated using the SHA-384 hash method.
SHA-512	The input value supplied in the data parameter is generated using the SHA-512 hash method.
Signature checking rule (One optional). Valid only with the PKCS-PSS digital signature hash formatting method.	
EXMATCH	Specifies that the 4-byte salt length prepended to the data identified by the data parameter must be an exact match to the salt length derived from the signature. This is the default.

NEXMATCH	Specifies that the 4-byte salt length prepended to the data identified by the data parameter does not have to be an exact match to the salt length derived from the signature. Note: The derived salt length cannot be less than the prepended length.
Trusted public key restriction (One, optional). Not valid with ECDSA, EDDSA, EC-SDSA, or CRDL-DSA keywords. Valid only with trusted blocks.	
TPK-ONLY	The PKA_public_key_identifier parameter must be a trusted block or the PKDS label of a trusted block that contains, at a minimum, two sections: • Trusted Block Information section 0x14, which is required for all trusted blocks and • Trusted Public Key section 0x11, which contains the trusted public key and usage rules that indicate whether the trusted public key can be used in digital signature operations.
Certificate validation method (One required when PKA_public_key_identifier is a certificate. Otherwise, must not be specified.)	
RFC-2459	Validate the certificate using the semantics of RFC-2459.
RFC-3280	Validate the certificate using the semantics of RFC-3280.
RFC-5280	Validate the certificate using the semantics of RFC-5280.
RFC-ANY	Attempt to validate the certificate by first using the semantics of RFC-2459, then the semantics of RFC-3280, and finally, the semantics of RFC-5280.
Public Key Infrastructure Usage (One optional when the input is an X.509 certificate. Otherwise, must not be specified.)	
PKI-CHK	Specifies that the X.509 certificate is to be validated against the trust chain of the PKI hosted in the adapter. This requires that the CA credentials have been installed into all coprocessors using the Trusted Key Entry workstation. This is the default.
PKI-NONE	Specifies that the X.509 certificate is not to be validated against the trust chain of the PKI hosted in the adapter. This is suitable if the certificate has been validated using host-based PKI services or if using a self-signed certificate.

PKA_public_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the public_key_identifier parameter. If the public_key_identifier contains a label, the value must be 64. Otherwise, the value must be between the actual length of the token and 6500.

PKA_public_key_identifier

Direction	Type
Input	String

The identifier of the public key to verify the signature. The key identifier is an RSA, EC, or CRYSTALSDilithium public key token, an internal trusted block, an X.509 certificate, or the label of such a public key identifier in key storage.

When the TPK-ONLY keyword is specified, the identifier must be a trusted block.

When the PKCS-PSS keyword is specified, the identifier cannot be an X.509 certificate.

Certificates may be PEM-formatted EBCDIC text or DER-encoded. The certificate may either have no key usage attribute or it must have at least one of the following usages: digitalSignature, nonRepudiation, keyCertSign, or cRLSign.

For the EC-SDSA algorithm, the key is restricted to secp256r1 (P256) and secp521r1 (P521) curves. The hash method for P256 keys is SHA-256. The hash method for P521 keys is SHA-512.

data_length

Direction	Type
Input	Integer

The length of the data parameter in bytes. See Usage Notes for details of the formatting methods and the length requirements.

When the data input type keyword is MESSAGE, the value is the length of the message to be hashed. The maximum value is 2147483647 bytes. When the signature algorithm keyword is EDDSA or EC-SDSA, the maximum length is 8192. When CRDL-DSA is specified, the maximum data_length is 5000. A hash method rule array keyword must be specified.

When the data input type keyword is HASH, the value must be the exact length of the hash to be signed. The maximum value is 516 bytes. The length of the hash for the supported hashing method is 16 for MD5, 20 for SHA-1 and RPMD-160, 28 for SHA-224, 32 for SHA-256, 48 for SHA-384, and 64 for SHA-512. You can use either the One-Way Hash Generate callable service or the MDC Generate callable service to generate the hash.

For the PKCS-PSS formatting rule, the first four bytes must be the salt length. The remaining bytes of the field must contain the hash or message. The value will be 4 + length of the hash or message.

data

Direction	Type
Input	String

The application-supplied text on which the supplied signature was generated. The data can be the message to be hashed and verified or the hash of the message. See Usage Notes for details of the formatting methods and the data requirements.

The data must be in the caller's primary address space.

For the PKCS-PSS formatting rule, the first four bytes must be the salt length. The remaining bytes of the parameter must contain the hash or message.

signature_field_length

Direction	Type
Input	Integer

The length in bytes of the signature_field parameter. The maximum size is 3500 bytes.

signature_field

Direction	Type
Input	String

This field contains the digital signature to verify. The digital signature is in the low-order bits (rightjustified) of a string whose length is the minimum number of bytes that can contain the digital signature. This string is left-justified within the signature_field.

Restrictions

The exponent for RSA public keys must be odd.

RSA keys 512-bit to 2048-bit may have a public exponent of 3, 5, 17, 257, 65537, or random. Support for RSA public exponents 5, 17, and 257 requires the October 2016 or later licensed internal code (LIC).

For 2049-bit to 4096-bit RSA keys:

- The public exponent may have a value of 3, 5, 17, 257, 65537, or random.
- Support for a random public exponent requires zEC12, zBC12, and later systems with a CEX4C or later coprocessor with September 2013 or later licensed internal code (LIC).
- Support for RSA public exponents 5, 17, and 257 requires the October 2016 or later licensed internal code (LIC).

Usage notes

SAF may be invoked to verify the caller is authorized to use this callable service, the key label, or internal secure key tokens that are stored in the CKDS or PKDS.

Notes on formatting the message:

ISO-9796

The length of the hash must be less than or equal to one-half of the number of bytes required to contain the modulus of the RSA key.

PKCS-1.0 and PKCS-1.1

The length of the hash must be 11 bytes shorter than the number of bytes required to contain the modulus of the RSA key.

When the HASH keyword is specified, the hash must be BER encoded. See “Formatting hashes and keys in public-key cryptography” on page 1472 for a description of the formatting methods.

PKCS-PSS

The first four bytes of the data parameter will contain the salt length. The remaining bytes of the parameter contains the hash or message.

The hash algorithm and salt length should be provided to you with the signature. If not, the recommended salt length is 0 or the byte length of the hash algorithm.

For signature verification, the salt length derived from the signature must be an exact match (keyword EXMATCH, the default) for the salt length specified with the data parameter. When the **Digital Signature Verify – PKCS-PSS allow not exact salt length** access control is enabled in the domain role, keyword NEXMATCH can be specified to allow signature verification when the salt length derived from the signature is not an exact

match for the salt length specified with the data parameter. Salt lengths derived from the signature are not allowed to be less than the value specified with the data parameter.

When the signature verification is done on an accelerator, there is no domain role and no access control points. The check of the salt length will be performed based on the signature check rule keywords.

X9.31

There are no restrictions for the hash length or message.

ZERO-PAD

The length of the hash must be less than or equal to the number of bytes required to contain the modulus of the RSA key.

ECDSA

There are no restrictions for the hash length or message.

EdDSA and EC-SDSA

The length of the message must be less than or equal to 8192 bytes.

CRDL-DSA

The length of the message must be less than or equal to 5000 bytes.

PKCS-PSS details

Before specifying PKCS-PSS, see section A.2.3 of RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2 (RFC 8017 (tools.ietf.org/html/rfc8017)). Section A.2.3 defines a parameter field for RSASSAPSS that has the following four parameters:

hashAlgorithm

This parameter identifies the hash function. A hashing-method specification keyword of SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512 is required. There is no default.

maskGenAlgorithm

This parameter identifies the mask generation function. MGF1 is a mask generation function based on a hash function and is the only function currently defined by the standard. For CCA, the hash function on which MGF1 is based is always the same as hashAlgorithm. Although this is not required, CCA enforces the recommendation by the standard that the underlying hash function be the same as the one identified by hashAlgorithm.

saltLength

This is the length of the salt, which is a randomly generated value. Use the data variable of the verb to prepend a 4-byte saltLength field in big endian format to the hash digest to be signed or the message to be hashed and signed.

trailerField

This is the trailer field number. This is not an input to the verb because it is always set to the value X'BC'. Other trailer field numbers are not supported by the standard.

Access control point

The **Digital Signature Verify** access control point controls the function of this service.

For the PKCS-PSS formatting method, the salt length derived from the signature must be an exact match for the salt length specified in the data parameter. When **the Digital Signature Verify – PKCS-PSS allow not exact salt length** access control is enabled in the domain role, the NEXMATCH keyword may be specified in the rule_array parameter. When the NEXMATCH keyword is specified, the salt length derived from the signature need not be an exact match for the salt length specified with the data parameter.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 69. Digital Signature Verify required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Keywords PKCS-PSS, EXMATCH, NEXMATCH, SHA-384, SHA-512, RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express2 Accelerator	RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.

IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Keywords PKCS-PSS, EXMATCH, NEXMATCH, SHA-384, SHA-512, RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express2 Accelerator	RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express3 Coprocessor	ECC support requires the Sept. 2010 licensed internal code (LIC). Keywords PKCS-PSS, EXMATCH, NEXMATCH, SHA-384, SHA-512, RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express3 Accelerator	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-

		DSA, and CRDLHASH are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	RSA clear key support with moduli within the range 2048-bit and 4096-bit requires the Sept. 2011 or later licensed internal code (LIC). Keywords PKCS-PSS, EXMATCH, NEXMATCH, SHA-384, SHA-512, RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express3 Accelerator	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Keywords PKCS-PSS, EXMATCH, NEXMATCH, SHA-384, SHA-512, RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. X.509 certificates are not supported.

		Compliant-tagged key tokens are not supported.
	Crypto Express3 Accelerator Crypto Express4 Accelerator	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	PKCS-PSS support requires the October 2016 or later licensed internal code (LIC). Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express5 Accelerator	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, CRDL-DSA, and CRDLHASH are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	PKCS-PSS support requires the October 2016 or later licensed internal code (LIC). Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported.

		ECC Koblitz curve secp256k1 is not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, and PKI-NONE require the July 2019 or later licensed internal code (LIC). Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC). Self-signed certificates require a CEX6C with the July 2019 or later licensed internal code (LIC). Keywords EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported.
	Crypto Express5 Accelerator Crypto Express6 Accelerator	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC Koblitz curve secp256k1 is not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Keywords EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported.

		ECC Koblitz curve secp256k1 is not supported.
	Crypto Express7 CCA Coprocessor	Keywords EDDSA, ED-HASH, CRDL-DSA, and CRDLHASH require the June 2020 or later licensed internal code (LIC). ECC Koblitz curve secp256k1 requires the September 2020 or later licensed internal code (LIC). Keyword EC-SDSA requires the CCA release 7.4 or later licensed internal code.
	Crypto Express5 Accelerator Crypto Express6 Accelerator Crypto Express7 Accelerator	Keywords RFC-2459, RFC-3280, RFC-5280, RFC-ANY, PKI-CHK, PKI-NONE, EDDSA, ED-HASH, EC-SDSA, CRDL-DSA, and CRDLHASH are not supported. ECC not supported. CRYSTALS-Dilithium not supported. X.509 certificates are not supported. Compliant-tagged key tokens are not supported.
	CP Assist for Cryptographic Functions	ECC requests for the Prime P-256, Prime P-384, Prime P-521 curves, Ed25519, and Ed448 are supported.

PKA Key Translate (CSNDPKT and CSNFPKT)

The PKA Key Translate callable service is used to do the following:

- Translation - Translate a CCA RSA key token into an external key token. The format of the external key token is specified by the output format keyword of the rule_array parameter. Supported output formats are: smart card, EMV, and PKCS #11 object.

  The source CCA RSA key token must be wrapped with a transport key-encrypting key (KEK). The XLATE bit must also be turned on in the key usage byte of the source token. The source token is unwrapped using the specified source transport KEK. The target key token will be wrapped with the specified target transport KEK. Existing information in the target token is overwritten. There are restrictions on which type key can be used for the source and target transport key tokens. These restrictions are enforced by access control points.

- Conversion
  - Convert the object protection key (OPK) in an CCA RSA private key token from a DES key to an AES key (EXTDWAKW, INTDWAKW)

The service will convert an existing internal or external RSA private key token. The modulus-exponent and Chinese Remainder Theorem forms are supported. Private key section identifiers 0x06, 0x08, and 0x09 can be converted.

If a format restriction keyword is specified, the output key token will have the format restriction flag in the associated data section of the token set to a requested value. The key token can only be used to create signatures in the format specified.

  - Change the key usage attributes of an internal CCA AES OPK key token (RSAAESC2 or RSAAESM2 private key). (INTUSCHG)
  - Convert an internal CCA key token to a compliant-tagged token. (COMP-TAG)
- Compliance checking – Check that a CCA key token can have the compliant tag.

The callable service name for AMODE(64) invocation is CSNFPKT.

Format

```
CALL CSNDPKT(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              source_key_identifier_length,
              source_key_identifier,
              source_transport_key_identifier_length,
              source_transport_key_identifier,
              target_transport_key_identifier_length,
              target_transport_key_identifier,
              target_key_token_length,
              target_key_token)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The data is identified in the exit_data parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. Value must be 1 or 2.

rule_array

Direction	Type
Input/Output	String

The rule_array contains keywords that provide control information to the callable service. See Table 437 on page 1025 for a list. A keyword is left-justified in an 8-byte field and padded on the right with blanks.

Table 70. Keywords for PKA Key Translate Rule Array

Keyword	Meaning
Process rule (one required)	
Translation rules The source key token must be an external token and allow translation (TRANSLAT/XLATE key usage). The target transport key must allow translation in the key usage attributes.	
CKM-RAKW	Specifies the translation of an external CCA key token containing an RSA or EC private key into an external encrypted PKCS #11 object. The target transport key is an RSA CCA public key to wrap the ephemeral AES key that wraps the private key in the object.
EMVCRT	Specifies the translation of an external CCA key token containing an RSA CRT private key into the EMVCRT format and wrapped using TDES-ECB.
EMVDDA	Specifies the translation of an external CCA key token containing an RSA CRT private key into EMV dynamic data authentication (DDA) format and wrapped using TDES-CBC.
EMVDDAE	Specifies the translation of an external CCA key token containing an RSA CRT private key into EMV DDAE format and wrapped using TDES-ECB.
SCCOMCRT	Specifies the translation of an external CCA key token containing an RSA CRT private key into the smart card Chinese Remainder Theorem format.
SCCOMME	Specifies the translation of an external CCA key token containing an RSA ME private key into the smart card Modulus-Exponent format.

SCVISA	Specifies the translation of an external CCA key token containing an RSA ME private key into the smart card Visa proprietary format.
Conversion rules	
COMP-TAG	Specifies to convert the internal PKA private key token into a compliant-tagged token. When the input key type is not RSAAESC2 or RSAAESM2, the following rules apply: • When key-usage KM-ONLY is enabled, the token will be converted to a compliant-tagged RSAAESC2 or RSAAESM2 with key-usage U-KEYENC enabled. • When key-usage SIG-ONLY is enabled, the token will be converted to a compliant-tagged RSAAESC2 or RSAAESM2 with key-usage U-DIGSIG enabled. • For all other key-usages, use INTUSCHG to set the desired usage prior to using COMP-TAG.
EXTDWAKW	Specifies to convert an external RSA private key token with a TDES wrapped OPK to an AESKW wrapped OPK (RSA-AESC, RSAAESC2, RSA-AESM, or RSAAESM2).
INTDWAKW	Specifies to convert an internal RSA private key token with a TDES wrapped OPK to an AESKW wrapped OPK (RSA-AESC, RSAAESC2, RSA-AESM, or RSAAESM2).
INTUSCHG	Specifies to change the usage attributes of an internal PKA key token. Requires keyword group PKA Key Usage Control. Not valid if the input key is compliant-tagged.
Compliant checking rules	
COMP-CHK	Checks if an internal PKA key token can have the compliant tag.
EMV DDA encrypted key part data format (one, optional). Only valid with output format keyword EMVDDA or EMVDDAE.	
EMV1	Original EMV DDA output format. This is the default.
EMVLENBT	Modified EMV DDA output format, which includes a length byte that becomes part of the encrypted key part section. The length byte, which replaces a post-padding byte of X'00' that EMV1 uses, is prepended to the clear key part. This length is valued to the number clear key-part bytes and does not include any pad bytes.
Format restriction (one, optional). Only valid with keywords INTDWAKW and EXTDWAKW.	
FR-NONE	Specifies to not restrict the private key to be used to a particular method. The key is usable for any method. This is the default.
FR-I9796	Specifies to render the private key usable with the ISO-9796 digital-signature hash formatting method.
FR-PK10	Specifies to render the private key usable with the PKCS-1.0 digital-signature hash formatting method.
FR-PK11	Specifies to render the private key usable with the PKCS-1.1 digital-signature hash formatting method.
FR-PSS	Specifies to render the private key usable with the PKCS-PSS digital-signature hash formatting method.
FR-X9.31	Specifies to render the private key usable with the X9.31 digitalsignature hash formatting method.
FR-ZPAD	Specifies to render the private key usable with the ZERO-PAD digital-signature hash formatting method.
PKA Key Usage Control. Only valid with INTUSCHG. The keywords specified reflect the only usage attributes that will be enabled in the output key token. All other usage attributes will be disabled.	
U-DIGSIG	Digital Signature usage is allowed.

	When input key type is RSAAESC2 or RSAAESM2, requires that the U-DIGSIG flag is enabled in the input key token. When input key type is not RSAAESC2 or RSAAESM2, requires that the KEY-MGMT or SIG-ONLY flag is enabled in the input key token.
U-NONRPD	Non-Repudiation usage is allowed. When input key type is RSAAESC2 or RSAAESM2, requires that the U-NONRPD flag is enabled in the input key token. When input key type is not RSAAESC2 or RSAAESM2, requires that the KEY-MGMT or SIG-ONLY flag is enabled in the input key token.
U-KCRTSN	keyCertSign usage is allowed. When input key type is RSAAESC2 or RSAAESM2, requires that the U- KCRTSN flag is enabled in the input key token. When input key type is not RSAAESC2 or RSAAESM2, requires that the KEY-MGMT or SIG -ONLY flag is enabled in the input key token.
U-CRLSN	Certificate Revocation List Sign usage is allowed. When input key type is RSAAESC2 or RSAAESM2, requires that the U-CRLSN flag is enabled in the input key token. When input key type is not RSAAESC2 or RSAAESM2, requires that the KEY-MGMT or SIG -ONLY flag is enabled in the input key token.
U-KEYENC	Key Encipherment usage is allowed. When input key type is RSAAESC2 or RSAAESM2, requires that the U- KEYENC flag is enabled in the input key token. When input key type is not RSAAESC2 or RSAAESM2, requires that the KEY-MGMT or KM -ONLY flag is enabled in the input key token.
U-DATENC	Data Encipherment usage is allowed. When input key type is RSAAESC2 or RSAAESM2, requires that the U- DATENC flag is enabled in the input key token. When input key type is not RSAAESC2 or RSAAESM2, requires that the KEY-MGMT or KM -ONLY flag is enabled in the input key token.
U-KEYAGR	Key agreement usage is allowed. When input key type is RSAAESC2 or RSAAESM2, requires that the U- KEYAGR flag is enabled in the input key token. When input key type is not RSAAESC2 or RSAAESM2, requires that the KEY-MGMT or KM -ONLY flag is enabled in the input key token.
Key Agreement Control (One, Optional). Only valid with U-KEYAGR.	
U-ENCONL	Only encipher operations are allowed during key agreement key establishment protocols. When input key type is RSAAESC2 or RSAAESM2, the U-DECONL must not be enabled in the input key token.

U-DECONL	Only decipher operations are allowed during key agreement key establishment protocols. When input key type is RSAAESC2 or RSAAESM2, the U-ENCONL must not be enabled in the input key token.

source_key_identifier_length

Direction	Type
Input	Integer

Length in bytes of the source_key_identifier parameter. If the source_key_identifier contains a label, the length must be 64. Otherwise, the value must be between the actual length of the token and 3500.

source_key_identifier

Direction	Type
Input	String

The key identifier of the EC or RSA private key to be processed. For translation, the key is an external key token wrapped with an AES or DES key-encrypting key. For OPK conversion, the token may be internal or external. External tokens are wrapped with a DES key encrypting key. When an internal token is specified, the transport keys are not used.

When keyword COMP-CHK is specified, this must be an internal RSA private key token.

When keyword COMP-TAG or INTUSCHG is specified, this must be an internal RSA private key token with private key section X'08', X'30', or X'31'.

When keyword CKM-RAKW is specified, this must be an external RSA private key token with private key section X'08', X'30', or X'31' or an external EC private key token with private key section X’20’. Compliance tagged key tokens are not supported.

source_transport_key_identifier_length

Direction	Type
Input	Integer

Length in bytes of the source_transport_key_identifier parameter. When the source_transport_key_identifier contains a label, the length must be 64. When the processing rule is INTDWAKW, INTUSCHG, COMP-CHK, or COMP-TAG, the value must be zero. Otherwise, the value must be between the actual length of the token and 725.

source_transport_key_identifier

Direction	Type
Input/Output	String

The key identifier of the key to unwrap the source key. The key identifier is an operational token or the key label of an operational token in key storage.

For EC and RSA RSA-AESC, RSAAESC2, RSA-AESM, or RSAAESM2 key tokens, this is an AES EXPORTER or IMPORTER key with the TRANSLAT key usage attribute. For other RSA key tokens, this is a DES EXPORTER or IMPORTER key with the XLATE

control vector attribute. See “Access control points” on page 1102 for details on the type of transport key that can be used.

When the source_transport_key_identifier_length is zero, this parameter is ignored.

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

target_transport_key_identifier_length

Direction	Type
Input	Integer

Length in bytes of the target_transport_key_identifier parameter. If the target_transport_key_identifier contains a label, the length must be 64. When the processing rule is INTDWAKW, INTUSCHG, COMP-CHK, or COMP-TAG, the value must be zero. Otherwise, the value must be between the actual length of the token and 3500.

target_transport_key_identifier

Direction	Type
Input/Output	String

The key identifier of the key that will wrap the output key in the target_key_token parameter. The key identifier is an operational token or the key label of an operational token in key storage.

When the processing rule is EMVCRT, EMVDDA, EMVDDAE, SCCOMCRT, SCCOMME, or SCVISA, the key is a DES IMPORTER or EXPORTER with the XLATE control vector attribute.

When the processing rule is EXTDWAKW, the key is an AES IMPORTER or EXPORTER with the TRANSLAT key usage attribute.

See “Access control points” on page 1102 for details on the type of transport key that can be used.

When the processing rule is CKM-RKAW, the key is an RSA public key with a modulus bit length of 2048, 3072, or 4096. The key will wrap the ephemeral AES key that wraps the private key.

When the target_transport_key_identifier_length is zero, this parameter is ignored.

If the token supplied was encrypted under the old master key, the token is returned encrypted under the current master key.

target_key_token_length

Direction	Type
Input/Output	Integer

Length in bytes of the target_key_token parameter. On output, the value in this variable is updated to contain the actual length of the target_key_token produced by the callable service. The maximum length is 3500 bytes.

If the COMP-CHK keyword is specified, this parameter must be 0.

target_key_token

Direction	Type
Output	String

The processed key token.

When converting to an AES OPK format, the token is a CCA key token wrapped by an AES key-encrypting key (EXTDWAKW) or an internal token (INTDWAKW). When the INTUSCHG keyword is specified, the output will be an internal RSA private key token with private key section X'30' and associated data version X'04' (RSAAESM2) or an internal RSA private key token with private key section X'31' and associated data version X'05' (RSAAESC2). Internal tokens may be stored in the PKDS.

When translating to a non-CCA format, the key token is wrapped with the key-encrypting key specified in the target_transport_key_identifier parameter. The key token is not a CCA token and cannot be stored in the PKDS.

When the processing rule CKM_RAKW, the output is a PKCS#11 structure containing the AES ephemeral key wrapped by the RSA public key specified in the target_transport_key_identifier parameter and the wrapped private key.

Restrictions

CCA RSA ME tokens will not be translated to the SCCOMCRT, EMV DDA, EMV DDAE, or the EMV CRT formats. CCA RSA CRT tokens will not be translated to the SCCOMME format. SCVISA only supports Modulus-Exponent (ME) keys.

The maximum modulus size of CCA RSA CRT tokens for the EMVDDA, EMVDDAE, or the EMVCRT formats is 2040 bits.

Only CCA RSA CRT tokens with a private section of X'08' are supported by the EMVDDA, EMVDDAE, or the EMVCRT rule array keywords.

Access control points

There are access control points that control use of the format rule array keywords and the type of transport keys that can be used.

Table 71. Required access control points for PKA Key Translate

Rule array keyword	Access control point
COMP-CHK	PKA Key Translate - Allow COMP-CHK
COMP-TAG	PKA Key Translate - Allow COMP-TAG
CKM-RAKW and the source key is an EC key	PKA Key Translate – From CCA ECC to CKM-RAKW format
CKM-RAKW and the source key is an RSA key	PKA Key Translate – From CCA RSA to CKM-RAKW format
EMVCRT	PKA Key Translate - from CCA RSA CRT to EMV CRT Format
EMVDDA	PKA Key Translate - from CCA RSA CRT to EMV DDA Format
EMVDDAE	PKA Key Translate - from CCA RSA CRT to EMV DDAE Format
EXTDWAKW	PKA Key Translate – Translate external key token
INTDWAKW	PKA Key Translate – Translate internal key token
INTUSCHG	PKA Key Translate – Allow INTUSCHG
SCCOMCRT	PKA Key Translate - from CCA RSA to SC CRT Format
SCCOMME	PKA Key Translate - from CCA RSA to SC ME Format
SCVISA	PKA Key Translate - from CCA RSA to SC Visa Format

These access control points control the key type combination shown in this table. One of these access control points must be enabled.

Table 72. Required access control points for source/target transport key combinations

Source transport key type	Target transport key type	Access control point
EXPORTER	EXPORTER	PKA Key Translate - from source EXP KEK to target EXP KEK
IMPORTER	EXPORTER	PKA Key Translate - from source IMP KEK to target EXP KEK
IMPORTER	IMPORTER	PKA Key Translate - from source IMP KEK to target IMP KEK
EXPORTER	IMPORTER	(Not allowed)

When the **Disallow translation from AES wrapping to DES wrapping** access control point is enabled, this service will fail if the source_transport_key_identifier is an AES key and the target_transport_key_identifier is a DES key.

When the **Disallow translation from AES wrapping to weaker AES wrapping** access control point is enabled, this service will fail if the source_transport_key_identifier is an AES key that is stronger than the target_transport_key_identifier.

When the **Disallow translation from DES wrapping to weaker DES wrapping** access control point is enabled, this service will fail if the source_transport_key_identifier is a DES key that is stronger than the target_transport_key_identifier.

The **Allow weak DES wrap of RSA** access control allows a weaker DES key-encrypting key to wrap an RSA private key token. The **Prohibit weak wrap – Transport keys** access control must be enabled and this access control will override the restriction.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service. The CCA releases used in the table are described in “CCA release levels,” on page 173.

Table 73. PKA Key Translate required hardware

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	Requires the April 2009 or later licensed internal code (LIC). The rule_array keywords CKM-RAKW, EMVDDA, EMVDDAE, EMVCRT, FR-NONE, FR-I9796, FR-PK10, FR-PK11, FR-PSS, FR-X9.31, FR-ZPAD, EMV1, and EMVLENBT are not supported. Triple-length DES keys are not supported. Key types RSAAESM2 and RSAAESC2 are not supported.

		Keywords COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL are not supported. Compliant-tagged key tokens are not supported.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor Crypto Express3 Coprocessor	Requires the April 2009 or later licensed internal code (LIC). The rule_array keywords CKM-RAKW, EMVDDA, EMVDDAE, EMVCRT, FR-NONE, FR-I9796, FR-PK10, FR-PK11, FR-PSS, FR-X9.31, FR-ZPAD, EMV1, and EMVLENBT are not supported. Triple-length DES keys are not supported. Key types RSAAESM2 and RSAAESC2 are not supported. Keywords COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL are not supported. Compliant-tagged key tokens are not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	Support for the rule_array keywords EMVDDA, EMVDDAE, and EMVCRT requires the March 2014 or later licensed internal code (LIC). The rule_array keywords CKM-RAKW, FR-NONE, FR-I9796, FRPK10, FR-PK11, FR-PSS, FR-X9.31, FR-ZPAD, EMV1, and EMVLENBT are not supported. Triple-length DES keys are not supported. Key types RSAAESM2 and RSAAESC2 are not supported. Keywords COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL are not supported. Compliant-tagged key tokens are not supported.

IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Support for the rule_array keywords EMVDDA, EMVDDAE, and EMVCRT requires the March 2014 or later licensed internal code (LIC). The rule_array keywords CKM-RAKW, FR-NONE, FR-I9796, FRPK10, FR-PK11, FR-PSS, FR-X9.31, FR-ZPAD, EMV1, and EMVLENBT are not supported. Triple-length DES keys are not supported. Key types RSAAESM2 and RSAAESC2 are not supported. Keywords COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL are not supported. Compliant-tagged key tokens are not supported
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	Support for the format restriction rule_array keywords requires the October 2016 or later licensed internal code (LIC). Triple-length DES keys and rule_array keywords EMV1 and EMVLENBT require the July 2019 or later licensed internal code (LIC). Key types RSAAESM2 and RSAAESC2 are not supported. Keywords CKM-RAKW, COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL are not supported. Compliant-tagged key tokens are not supported.
IBM z14 IBM z14 ZR1	Crypto Express5 CCA Coprocessor	Support for the format restriction rule_array keywords requires the October 2016 or later licensed internal code (LIC). Triple-length DES keys and rule_array keywords EMV1 and EMVLENBT require the July 2019 or later licensed internal code (LIC).

		Key types RSAAESM2 and RSAAESC2 are not supported. Keywords CKM-RAKW, COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Triple-length DES keys and rule_array keywords EMV1 and EMVLENBT require the December 2018 or later licensed internal code (LIC). Key types RSAAESM2 and RSAAESC2 require the July 2019 or later licensed internal code (LIC). Keywords CKM-RAKW, COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL require the July 2019 or later licensed internal code (LIC). Compliant-tagged key tokens require a CEX6C with the July 2019 or later licensed internal code (LIC).
IBM z15 IBM z15 TO2	Crypto Express5 CCA Coprocessor	Key types RSAAESM2 and RSAAESC2 are not supported. Keywords CKM-RAKW, COMP-CHK, COMP-TAG, INTUSCHG, U-DIGSIG, U-NONRPD, U-KCRTSN, U-CRLSN, U-KEYENC, U-DATENC, U-KEYAGR, U-ENCONL, and U-DECONL are not supported. Compliant-tagged key tokens are not supported.
	Crypto Express6 CCA Coprocessor	Rule array keyword CKM-RAKW is not supported.
	Crypto Express7 CCA Coprocessor	Rule array keyword CKM-RAKW requires the CCA release 7.4 or later licensed internal code (LIC).

ICSF Query Facility2 (CSFIQF2 and CSFIQF26)

Use this utility to retrieve status information about the cryptographic environment as currently known to ICSF.

This callable service will:

- NOT be SAF protected.
- NOT make calls to any cryptographic processor
- Return information that can be collected from various ICSF control blocks

The callable service name for AMODE(64) invocation is CSFIQF26..

Format

```
CALL CSFIQF2(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              returned_data_length,
              returned_data,
              reserved_data_length,
              reserved_data)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1279 lists the reason codes.

exit_data_length

Direction	Type
Ignored	Integer

This field is ignored. It is recommended to specify 0 for this parameter.

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you are supplying in rule_array. Value must be 0.

rule_array

Direction	Type
Ignored	String

Keywords that provide control information to callable services. This parameter is ignored.

returned_data_length

Direction	Type
Input/Output	Integer

The length of the returned_data parameter in bytes. A minimum value of 11 is required. Specify a length of 22 or greater to receive all the supported data.

returned_data

Direction	Type
Output	String /Integer

This field will contain the output from the service. The service will return only the amount of data specified by the returned_data_length field.

The format of the returned_data is defined in Table 504 on page 1164.

Table 74. Format of returned ICSF Query Facility 2 data

Bytes	Description
0-7	ICSF FMID.
8	**Bit** **Meaning when set on** 0 Crypto Accelerator available. 1 CCA Coprocessor available. 2 Public Key hardware available. 3 TKDS available. 4 SHA-1 available in CPACF. 5 SHA-224 available in CPACF. 6

	SHA-256 available in CPACF. 7 SHA-384 available in CPACF.
9	**Bit** **Meaning when set on** 0 SHA-512 available in CPACF. 1 DES available in CPACF. 2 TDES available in CPACF. 3 AES 128-bit available in CPACF. 4 AES 192-bit available in CPACF. 5 AES 256-bit available in CPACF. 6 AES-GCM available in CPACF. 7 ECC Clear Key hardware available.
10	**Bit** **Meaning when set on** 0 ECC Secure Key hardware available. 1 PKCS #11 Secure Key available. 2 FIPS No Enforcement Mode. 3 FIPS Mode enabled. 4 FIPS Compatibility Mode enabled. 5 Reserved. 6 SHA-3 and SHAKE available. 7 ECC available in CPACF.
11	Reserved.
12-19	System compliance information. Byte 0 **Bit** **Meaning when set on** 0 Compliance mode is active. 1 Compliance migration mode is active. 2-7 Reserved. Bytes 1-6: Reserved. Byte 7 **Bit** **Meaning when set on**

	0-6 Reserved. 7 PCI-HSM 2016 compliance mode is active. Note: These byte references only relate to the 8-byte structure contained in bytes 12-19.
20	Crypto usage statistics flags. **Bit** **Meaning when set on** 0 Cryptographic engine usage tracking is enabled (ENG). 1 Cryptographic service usage tracking is enabled (SRV). 2 Cryptographic algorithm usage tracking is enabled (ALG). 3-7 Reserved.
21	Supported elliptic curves. **Value (hex)** **Supported curves** 01 secp192r1, secp224r1, secp256r1, secp384r1, secp521r1, brainpoolP160r1, brainpoolP192r1, brainpoolP224r1, brainpoolP256r1, brainpoolP320r1, brainpoolP384r1, brainpoolP512r1. 02 All of the above curves plus: Ed25519, X25519, Ed448, X448, secp256k1 03 All of the above curves plus: Koblitz secp256k1. Note that some curves require hardware.

reserved_data_length

Direction	Type
Input	Integer

The length of the reserved_data parameter. This field is reserved and must be 0.

reserved_data

Direction	Type
Ignored	String

This parameter is ignored.

Required hardware

No cryptographic hardware is required by this callable service.

PKCS #11 Private Key Sign (CSFPPKS and CSFPPKS6)

Use the PKCS #11 Private Key Sign callable service to:

- Decrypt or sign data using an RSA private key using zero-pad or PKCS #1 v1.5 formatting.
- Sign data using a CRYSTALS-Dilithium (LI2) private key.
- Sign data using a DSA private key.
- Sign data using an Elliptic Curve private key in combination with DSA.

The key handle must be a handle of a PKCS #11 private key object. When the request type keyword DECRYPT is specified in the rule array, CKA_DECRYPT attribute must be true. When no request type is specified, the CKA_SIGN attribute must be true.

The callable service can be invoked in AMODE(24), AMODE(31), or AMODE(64). 64-bit callers must use CSFPPKS6.

Format

```
CALL CSFPPKS(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              cipher_value_length,
              cipher_value,
              key_handle,
              clear_value_length,
              clear_value )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the return codes.

reason_code

Direction	Type
Output	String

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the reason codes.

exit_data_length

Direction	Type

Ignored	Integer

This field is ignored. It is recommended to specify 0 for this parameter.

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array_parameter. This value may be 1 or 2.

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service.

Table 75. Keywords for private key sign

Keyword	Meaning
Mechanism (One of the following must be specified)	
DSA	Mechanism is DSA signature generation
ECDSA	Mechanism is Elliptic Curve with DSA signature generation
EC-SDSA	Mechanism is non-ECDSA non-EDDSA EC-based signature, generally Schnorr variants.
EDDSA	Mechanism is PureEdDSA signature generation (no pre-hashing)
LI2	Mechanism is a CRYSTALS-Dilithium signature generation.
RSA-PKCS	Mechanism is RSA decryption or signature generation using PKCS #1 v1.5 formatting
RSA-ZERO	Mechanism is RSA decryption or signature generation using zero-pad formatting
Request type (optional)	
DECRYPT	The request is to decrypt data. This type of request requires the CKA_DECRYPT attribute to be true. If DECRYPT is not specified, the CKA_SIGN attribute must be true. Valid with RSA only.
Schnorr Subvariant (One, required with EC-SDSA)	
RANDOM	Randomized Schnorr signature; no pre-hashing, SHA-256 only
COMPMULT	Randomized Schnorr signature with compressed keys, and including the signing party's public key, SHA-256 only

cipher_value_length

Direction	Type
Input	Integer

Length of the cipher_value parameter in bytes.

cipher_value

Direction	Type
Input	String

For decrypt, this is the value to be decrypted. Otherwise, this is the value to be signed.

- For DSA and ECDSA signature requests, the data to be signed is expected to be a SHA1, SHA224, SHA256, SHA384, or SHA512 digest.
- For EC-SDSA signature requests, the data to be signed is expected to be a SHA1, SHA224, or SHA256 digest.
- For CRYSTALS-Dilithium signature requests,
  - The data to be signed is from zero to 5120 bytes.
- For EDDSA signature requests,
  - When using a clear key, the data to be signed is from zero to 214 (16384) bytes.
  - When using a secure key, the data to be signed is from zero to 213 (8192) bytes.
- For RSA-PKCS signature requests, the data to be signed is expected to be a DER encoded DigestInfo structure.

key_handle

Direction	Type
Input	String

The 44-byte handle of a private key object. See “Handles” on page 103 for the format of a key_handle.

clear_value_length

Direction	Type
Input/Output	Integer

Length of the clear_value parameter in bytes. On input, this must be at least the size of the RSA modulus in bytes. For CRYSTALS-Dilithium signatures, this must be at least 3366 bytes. On output, this is updated to be the actual length of the decrypted value or the generated signature.

clear_value

Direction	Type
Output	String

For decrypt, this field will contain the decrypted value. Otherwise this field will contain the generated signature.

Authorization

To use this service with a public object, the caller must have SO (READ) authority or USER (READ) authority (any access).

To use this service with a private object, the caller must have USER (READ) authority (user access).

Usage Notes

Operations may be done in hardware or software.

Request type DECRYPT is not supported for an Elliptic Curve, DSA, or CRYSTALS-Dilithium private key.

For rule EC-SDSA and each subvariant, the signing key must be a secure key.

PKCS #11 Public Key Verify (CSFPPKV and CSFPPKV6)

Use the PKCS #11 Public Key Verify callable service to:

- Encrypt or verify data using an RSA public key using zero-pad or PKCS #1 v1.5 formatting. For encryption, the encrypted data is returned.
- Verify a signature using a CRYSTALS-Dilithium public key. No data is returned.
- Verify a signature using a DSA public key. No data is returned.
- Verify a signature using an Elliptic Curve public key in combination with DSA. No data is returned.

The key handle must be a handle of a PKCS #11 public key object. When the request type keyword ENCRYPT is specified in the rule array, CKA_ENCRYPT attribute must be true. When no request type is specified, the CKA_VERIFY attribute must be true.

The callable service can be invoked in AMODE(24), AMODE(31), or AMODE(64). 64-bit callers must use CSFPPKV6.

Format

```
CALL CSFPPKV(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              clear_value_length,
              clear_value,
              key_handle,
              cipher_value_length,
              cipher_value )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. Appendix A, “ICSF and cryptographic coprocessor return and reason codes,” on page 1283 lists the reason codes.

exit_data_length

Direction	Type
Ignored	Integer

This field is ignored. It is recommended to specify 0 for this parameter.

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array parameter. This value must be 1 or 2.

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service.

Table 76. Keywords for public key verify

Keyword	Meaning
Mechanism (One of the following must be specified)	
DSA	Mechanism is DSA signature verification
ECDSA	Mechanism is Elliptic Curve with DSA signature verification
EC-SDSA	Mechanism is non-ECDSA non-EDDSA EC-based signature, generally Schnorr variants.
EDDSA	Mechanism is PureEdDSA signature verification (no pre-hashing)
LI2	Mechanism is a CRYSTALS-Dilithium signature verification.
RSA-PKCS	Mechanism is RSA encryption or signature verification using PKCS #1 v1.5 formatting
RSA-ZERO	Mechanism is RSA encryption or signature verification using zero-pad formatting
Request type (optional)	
ENCRYPT	The request is to encrypt data. This type of request requires the CKA_ENCRYPT attribute to be true. If ENCRYPT is not specified, the CKA_VERIFY attribute must be true. Valid with RSA only.
Schnorr Subvariant (One, required with EC-SDSA)	
RANDOM	Randomized Schnorr signature; no pre-hashing, SHA-256 only.

COMPMUL	Randomized Schnorr signature with compressed keys, and including the signing party's public key, SHA-256 only

clear_value_length

Direction	Type
Input	Integer

The length of the clear_value parameter

clear_value

Direction	Type
Input	String

For encrypt, this is the value to be encrypted. Otherwise, this is the signature to be verified.

key_handle

Direction	Type
Input	String

The 44-byte handle of public key object. See “Handles” on page 103 for the format of a key_handle.

cipher_value_length

Direction	Type
Input/Output	Integer

For encrypt, on input, this is the length of the cipher_value parameter in bytes. On output, this is updated to be the actual length of the text encrypted into the cipher_value parameter. For signature verification, this is the length of the data to be verified (input only).

cipher_value

Direction	Type
Input	String

For encrypt, this is the encrypted value (output only).

- For CRYSTALS-Dilithium signature verification requests,
  - The data to be verified is from zero to 5120 bytes.
- For DSA and ECDSA signature verification requests, the data to be verified is expected to be a SHA1, SHA224, SHA256, SHA384, or SHA512 digest.
- For EC-SDSA signature verification requests, the data to be verified is expected to be a SHA1, SHA224, or SHA256 digest.
- For EDDSA signature requests,
  - When using a clear key, the data to be verified is from zero to 214 (16384) bytes.
  - When using a secure key, the data to be verified is from zero to 213 (8192) bytes.
- For RSA-PKCS signature verification requests, the data to be verified is expected to be a DER encoded DigestInfo structure.

- For signature verification, this is the data to be verified (input only).

Authorization

To use this service with a public object, the caller must have SO (READ) authority or USER (READ) authority (any access).

To use this service with a private object, the caller must have USER (READ) authority (user access).

Usage Notes

Operations may be done in hardware or software.

Request type ENCRYPT is not supported for an Elliptic Curve, DSA, or CRYSTALS-Dilithium public key.

For rule EC-SDSA and each subvariant, the verifying key must be a secure key.

PKCS #11 Secret Key Reencrypt (CSFPSKR and CSFPSKR6)

Use the PKCS #11 Secret Key Reencrypt callable service to decrypt data and then reencrypt the data using secure secret keys. The interim clear text created by the decrypt process is never available to the application and never exists outside of the EP11 coprocessor. AES and DES3 secure keys are supported. CBC, CBC-PAD, and ECB modes are supported.

Both key handles must be handles of a PKCS #11 secure secret key object. The CKA_DECRYPT attribute must be true for the decrypt key. The CKA_ENCRYPT attribute must be true for the encrypt key.

If the length of output field is too short to hold the output, the service will fail and return the required length of the output field in the encrypted_text_length parameter.

The callable service can be invoked in AMODE(24), AMODE(31), or AMODE(64). 64-bit callers must use CSFPSKR6.

Format

```
CALL CSFPSKR(
            return_code,
            reason_code,
            exit_data_length,
            exit_data,
            rule_array_count,
            rule_array,
            decrypt_handle,
            encrypt_handle,
            decrypt_initialization_vector_length,
            decrypt_initialization_vector,
            encrypt_initialization_vector_length,
            encrypt_initialization_vector,
            chain_data_length,
            chain_data,
```

```
decrypt_text_length,
decrypt_text,
decrypt_text_id,
encrypt_text_length,
encrypt_text,
encrypt_text_id)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. See appendix A, “ICSF and cryptographic coprocessor return and reason codes” in the z/OS Cryptographic Services ICSF Application Programmer’s Guide.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. See appendix A, “ICSF and cryptographic coprocessor return and reason codes” in the z/OS Cryptographic Services ICSF Application Programmer’s Guide.

exit_data_length

Direction	Type
Ignored	Integer

This field is ignored. It is recommended to specify 0 for this parameter.

exit_data

Direction	Type
Ignored	String

This field is ignored.

Rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the rule_array_parameter. This value must be 2.

rule_array

Direction	Type
Input	Integer

Keywords that apply to the decryption of the decrypt_text and the encryption of the interim clear text.

Table 77. Rule Array Keywords for rule_array	
Keyword	**Meaning**
Decrypt Processing rule (one required)	
D-CBC	Performs cipher block chaining on the decrypt_text. The decrypt_text_length must be a multiple of the block size for the specified encrypt mechanism (8 bytes for DES3 and 16 bytes for AES).
D-CBCPAD	Performs cipher block chaining on the decrypt_text. The decrypt_text_length must be a multiple of the block size for the specified decrypt mechanism (8 bytes for DES3 and 16 bytes for AES).
D-ECB	Performs electronic code book encryption. The decrypt_text_length must be a multiple of the block size for the specified decrypt mechanism (8 bytes for DES3 and 16 bytes for AES).
Encrypt Processing rule (one required)	
E-CBC	Performs cipher block chaining on the interim clear text. The interim clear text length must be a multiple of the block size for the specified encrypt mechanism (8 bytes for DES3 and 16 bytes for AES).
E-CBCPAD	Performs cipher block chaining on the interim clear text. The interim clear text length may be shorter than the block size for the encrypt mechanism or may even be zero. PKCS #7 padding is performed, thus the encrypt_text_length will be greater than the interim clear text length
E-ECB	Performs electronic code book encryption. The interim clear text length must be a multiple of the block size for the specified encrypt mechanism (8 bytes for DES3 and 16 bytes for AES).

decrypt_handle

Direction	**Type**
Input	String

The 44-byte handle of the secure secret key object used to decrypt the decrypt_text.

encrypt_handle

Direction	**Type**
Input	String

The 44-byte handle of the secure secret key object used to encrypt the interim clear text.

decrypt_initialization_vector_length

Direction	**Type**
Input	Integer

Length of the decrypt_initialization_vector in bytes. For CBC and CBC-PAD this must be 8 bytes for DES and 16 bytes for AES. For ECB this must be zero.

decrypt_initialization_vector

Direction	Type
Input	String

This is the 8 byte or 16 byte initial chaining value used for the decryption of the decrypt_text.

encrypt_initialization_vector_length

Direction	Type
Input	Integer

Length of the encrypt_initialization_vector in bytes. For CBC and CBC-PAD this must be 8 bytes for DES and 16 bytes for AES. For ECB this must be zero.

encrypt_initialization_vector

Direction	Type
Input	String

This is the 8 byte or 16 byte initial chaining value used for the encryption of the interim clear text.

chain_data_length

Direction	Type
Input	Integer

This value must be zero.

chain_data

Direction	Type
Input/Output	String

This parameter is ignored when chain_data_length is zero.

decrypt_text_length

Direction	Type
Input	Integer

The length of the decrypt_text parameter in bytes. The length can be up to 10600.

decrypt_text

Direction	Type
Input	String

Text to be decrypted and then encrypted.

decrypt_text_id

Direction	Type
Input	Integer

The ALET identifying the space where the decrypt_text resides.

encrypt_text_length

Direction	Type
Input/Output	Integer

On input, the length in bytes of the encrypt_text parameter. On output, the length of the text reencrypted into the encrypt text parameter.

encrypt_text

Direction	Type
Output	String

The encrypted text resulting from the decryption and reencryption of the decrypt_text.

encrypt_text_id

Direction	Type
Input	Integer

The ALET identifying the space where the encrypt_text resides.

Authorization

To use this service with a public object, the caller must have at least SO (READ) authority or USER (READ) authority (any access).

To use this service with a private object, the caller must have at least USER (READ) authority (user access).

Usage Notes

The use of this service keys requires an active EP11 Coprocessor. If there is not an EP11 Coprocessor online that supports Secret Key Reencrypt, the service will return with reason code ‘C’x, return code ‘2B34’x.

ICSF and cryptographic coprocessor return and reason codes

Return codes and reason codes

Reason codes for return code 8 (8)

Table 78. Reason codes for return code 8 (8)

Reason Code Hex (Decimal)	Description
8FA (2298)	The hash function specified in the rule array has a digest size less than the bit length of the curve of the key. **User action**: Select a hash function large enough for the curve.
9D2 (2514)	An error was found in the ISO PIN block format. The specific error is not noted.

	User action: Examine the PIN profile, PAN data, and other input data to ensure the inputs are correct.
DD6 (3542)	The value specified in the input_PAN_data_length, PAN_data_length, or reference_PAN_data_length parameter is not valid. **User action**: Correct the input_PAN_data_length, PAN_data_length or reference_PAN_data_length parameter.
DD8 (3544)	The value specified in the input_PIN_profile_length or reference_PIN_profile_length parameter is not valid. **User action**: Correct the input_PIN_profile_length or reference_PIN_profile_length parameter.

Access control points and callable services

Table 79. Access control points affecting multiple services or requiring special consideration

Name	Callable services	Notes	Value (hex)	Usage
ANSI X9.8 PIN - Allow only ANSI PIN blocks	CSNBPTR / CSNEPTR, CSNBPTR2 / CSNEPTR2, CSNBPTRE / CSNEPTRE, CSNBPVR2 / CSNEPVR2, CSNBSPN / CSNESPN	See “ANSI X9.8 PIN restrictions” for a description of this control.	0352	DD, SC
Enhanced PIN Security	CSNBCPE / CSNECPE, CSNBCPA / CSNECPA, CSNBEPG / CSNEEPG, CSNBPTR / CSNEPTR, CSNBPTR2 / CSNEPTR2, CSNBPTRE / CSNEPTRE, CSNBPVR / CSNEPVR, CSNBPVR2 / CSNEPVR2, CSNBPCU / CSNEPCU, CSNBPFO / CSNEPFO	See “Enhanced PIN security mode” on page 79 for a description of this control.	0313	DD, SC
General ISO PIN Error Mode	CSNBPTR / CSNEPTR, CSNBPTR2 /	See “PIN block error processing mode” on page 80 for a description of this control.	039F	DD, SC

	CSNEPTR2, CSNBDPC / CSNEDPC, CSNBDPV / CSNEDPV			
Encrypted PIN Translate - Translate PIN Check Mode	CSNBPTR / CSNEPTR, CSNBPTR2 / CSNEPTR2	See “Enhanced PIN checking for CSNBPTR and CSNBPTR2” on page 80 for a description of this control.	03A0	DD, SC

Table 80. Access control points – Callable Services

Name	Callable service	Offset (Hex)	Usage
Authentication Parameter Generate - Clear	CSNBAPG	02B2	ED
Diversified Key Generate - A28XOREC	CSNBDKG	03B9	ED
Diversified Key Generate - A28OWFCL	CSNBDKG	03BA	ED
Diversified Key Generate - A28OWFEC	CSNBDKG	03BB	ED
Encrypted PIN Verify2 – REFPIN	CSNBPVR2	03B0	ED
Encrypted PIN Verify2 – TRUNCPIN	CSNBPVR2	03B1	ED
PKA Key Translate – From CCA RSA to CKM-RAKW format	CSNDPKT	03B6	DD
PKA Key Translate – From CCA ECC to CKM-RAKW format	CSNDPKT	03B7	DD
Random Number Generate Long – TDES-CBC	CSNBRNGL	03B5	ED
Symmetric Algorithm Encipher – allow A28MACGN and A28MACVR	CSNBSAE	03B2	ED
Symmetric Algorithm Encipher - allow A28OWFCL	CSNBSAE	02B3	ED
Symmetric Algorithm Encipher – allow A28OWFEC	CSNBSAE	02B4	ED
Symmetric Key Export – AES, CKM-RAKW	CSNDSYX	03B8	DD

Resource names for CCA and ICSF entry points

Table 81. Resource names for CCA and ICSF entry points

Descriptive service name	CCA entry point name		ICSF entry point name		SAF resource name	Callable service exit name
Encrypted PIN Verfiy2	CSNBPVR2	CSNBEPVR2	CSFPVR2	CSFPVR26	Encrypted PIN Verfiy2	CSNBPVR2
PKCS #11 Secret Key Reencrypt	N/A	N/A	CSFPSKR	CSFPSKR6	PKCS #11 Secret Key Reencrypt	N/A

Notes

– [1] CSF1xxx is just another name for the CSFPxxx service.

CCA release levels

Table 82. CCA release levels for the IBM z15

CCA Release	ICSF release APAR	Crypto Express adapter	Licensed internal code information
7.4	HCR77D1 OA61253	CEX7C	September 2021 Driver D41C MCL P46646.017
6.7	HCR77D1 OA61253	CEX6C	September 2021 Driver D41C MCL P46644.012
7.3	HCR77D1 OA60318	CEX7C	May 2021 Driver D41C MCL P46646.014
6.6	HCR77D1 OA60318	CEX6C	April 2021 Driver D41C MCL P46644.010
5.7	HCR77D1 OA60318	CEX5C	April 2021 Driver D41C MCL P46642.009
7.2	HCR77D1 OA59593	CEX7C	September 2020 Driver D41C MCL P46646.011
6.5	HCR77D1 OA59593	CEX6C	September 2020 Driver D41C MCL P46644.007
7.1	HCR77C1 OA58880	CEX7C	June 2020 Driver D41C MCL P46646.008
6.4	HCR77C1 OA58880	CEX6C	June 2020 Driver D41C MCL P46644.006
5.6	HCR77C1 OA58880	CEX5C	June 2020 Driver D41C MCL P46642.005
7.0	HCR77C1 OA58306	CEX7C	November 2019 Driver D41C MCL P46646.004
6.3	HCR77C1 OA58306	CEX6C	November 2019 Driver D36C MCL P41456.005 P41456.006
5.5	HCR77C1 OA58306	CEX5C	November 2019 Driver D41C MCL P46642.003
7.0	HCR77D1 z/OS V2R2-V2R4	CEX7C	September 2019
6.3	HCR77D1 z/OS V2R2-V2R4	CEX6C	September 2019
5.5	HCR77D1 z/OS V2R2-V2R4	CEX5C	September 2019

Table 83. CCA release levels for the IBM z14

CCA release	ICSF release APAR	Crypto Express adapter	Licensed internal code information
6.7	HCR77D1 OA61253	CEX6C	September 2021 Driver D36C MCL P41458.012
6.6	HCR77D1 OA60318	CEX6C	July 2021 Driver D36C MCL P41458.011
5.7	HCR77D1 OA60318	CEX5C	July 2021 Driver D36C MCL P41456.010
6.5	HCR77D1 OA59593	CEX6C	October 2020 Driver 36C MCL P41458.010

	OA60355		
6.4	HCR77C1 OA58880	CEX6C	June 2020 Driver D36C MCL P41458.009
5.6	HCR77C1 OA58880	CEX5C	June 2020 Driver D36C MCL P41456.008
6.3	HCR77C1 OA58306	CEX6C	November 2019 Driver D41C MCL P46644.003
5.5	HCR77C1 OA58306	CEX5C	November 2019 Driver D36C MCL P41456.005 P41456.006
6.3	HCR77D0 OA57089	CEX6C	July 2019 Driver D36C MCL P41458.004
5.5	HCR77D0 OA57089	CEX5C	July 2019 Driver D36C MCL P41456.004
6.3	HCR77D0 OA57088	CEX6C	July 2019 Driver D36C MCL P41458.004
5.5	HCR77D0 OA57088	CEX5C	July 2019 Driver D36C MCL P41456.004
6.2	HCR77D0 z/OS V2R2-V2R4	CEX6C	December 2018
6.1	HCR77C1 OA55184	CEX5C	December 2018 Driver D36C MCL P41458.002
5.4	HCR77C1 OA55184	CEX5C	December 2018 Driver D32L MCL P42641.004
6.0	HCR77C1 z/OS V2R1-V2R3	CEX6C	September 2017

Table 84. CCA release levels for the IBM z13

CCA release	ICSF release APAR	Crypto Express adapter	Licensed internal code information
5.7	HCR77D1 OA60318	CEX5C	April 2021 Driver D27I MCL P08449.024
5.6	HCR77C1 OA58880	CEX5C	June 2020 Driver D27I MCL P08449.022
5.5	HCR77C1 OA58306	CEX5C	November 2019 Driver D27I MCL P08449.020
5.5	HCR77D0 OA57089	CEX5C	July 2019 Driver D27I MCL P08449.019
5.5	HCR77D0 OA57088	CEX5C	July 2019 Driver D27I MCL P08449.019
5.4	HCR77C1 OA55184	CEX5C	December 2018 Driver D27I MCL P08449.019
5.3	HCR77C0 z/OS V2R1-V2R3	CEX5C	October 2016
5.2	HCR77B1 z/OS V1R13-V2R2	CEX5C	March 2016
5.1	HCR77B1 OA49064	CEX5C	July 2015
5.0	HCR77B0 z/OS V1R13-V2R2	CEX5C	February 2015

Chapter 6. Update of z/OS Cryptographic Services ICSF Writing PKCS #11 Applications, SC14-7510-08, information

This topic contains updates to the document z/OS Cryptographic Services ICSF Writing PKCS #11 Applications, SC14-7510-07, for the updates provided by this APAR. Refer to this source document if background information is needed.

Key types and mechanisms supported

Table 85. Mechanism information as returned by C_GetMechanismInfo (CK_MECHANISM_INFO)

Footnotes for Table 4 on page 21

1. The PKCS #11 standard designates two ways of implementing Elliptic Curve Cryptography, which is nicknamed F_p and F_{2^m}. z/OS PKCS #11 supports the F_p variety only.
2. ANSI X9.62 has the following ASN.1 definition for Elliptic Curve domain parameters:

```
Parameters ::= CHOICE {
      ecParameters  ECParameters,
      namedCurve    OBJECT IDENTIFIER,
      implicitlyCA  NULL }
```

z/OS PKCS #11 supports the specification of CKA_EC_PARAMS attribute by using the namedCurved CHOICE. The following NIST-recommended named curves are supported:

- secp192r1 – { 1 2 840 10045 3 1 1 }
- secp224r1 – { 1 3 132 0 33 }
- secp256r1 – { 1 2 840 10045 3 1 7 }
- secp384r1 – { 1 3 132 0 34 }
- secp521r1 – { 1 3 132 0 35 }

The following Brainpool-defined named curves are supported:

- brainpoolP160r1 – { 1 3 36 3 3 2 8 1 1 1 }
- brainpoolP192r1 – { 1 3 36 3 3 2 8 1 1 3 }
- brainpoolP224r1 – { 1 3 36 3 3 2 8 1 1 5 }
- brainpoolP256r1 – { 1 3 36 3 3 2 8 1 1 7 }
- brainpoolP320r1 – { 1 3 36 3 3 2 8 1 1 9 }
- brainpoolP384r1 – { 1 3 36 3 3 2 8 1 1 11 }
- brainpoolP512r1 – { 1 3 36 3 3 2 8 1 1 13 }

The following Edwards named curves are supported:

- Ed448 – { 1 3 101 113 }
- Ed25519 – { 1 3 101 112 }

The following Montgomery named curves are supported:

- X448 – { 1 3 101 111 }
- X25519 – { 1 3 101 110 }

The following Koblitz named curves are supported:

- secp256k1 - { 1 3 132 0 10 }

The following table lists the mechanisms supported by specific cryptographic hardware. When a particular mechanism is not available in hardware, ICSF uses the software implementation of the mechanism.

Table 86. Mechanisms supported by specific cryptographic hardware

Machine type and cryptographic hardware	Mechanisms supported	Notes
IBM z13 or z13s – CEX5P	CKM_DES_KEY_GEN CKM_DES2_KEY_GEN CKM_DES3_KEY_GEN CKM_RSA_PKCS CKM_RSA_X_509 CKM_MD5_RSA_PKCS CKM_SHA1_RSA_PKCS CKM_DES_CBC CKM_DES_CBC_PAD CKM_DES3_CBC CKM_DES3_CBC_PAD CKM_SHA_1 CKM_BLOWFISH_KEY_GEN CKM_RC4_KEY_GEN CKM_AES_KEY_GEN CKM_SSL3_PRE_MASTER_KEY_GEN CKM_TLS_PRE_MASTER_KEY_GEN CKM_GENERIC_SECRET_KEY_GEN CKM_RSA_PKCS_KEY_PAIR_GEN CKM_EC_KEY_PAIR_GEN CKM_DES_ECB CKM_DES3_ECB CKM_RSA_PKCS_KEY_PAIR_GEN CKM_DES_ECB CKM_DES3_ECB CKM_SHA224_RSA_PKCS CKM_SHA256_RSA_PKCS CKM_SHA224 CKM_SHA256 CKM_AES_CBC CKM_AES_CBC_PAD CKM_AES_CTS CKM_AES_ECB CKM_PKCS5_PBKD2	This is the base set
IBM z14 or z14 ZR - CEX5P	z13 and z13s set	
IBM z14 or z14 ZR - CEX6P	z14 and z14R1 set and • ReencryptSingle function • CKM_IBM_ECDSA_OTHER	CEX6P at level 3.6.16 is required for Reencrypt Single. CEX6P at level 3.6.19 is required for CKM_IBM_ECDSA_OTHER.
IBM z15 T01 or z15 T02 - CEX6P	IBM z14 or z14R1 set and • ReencryptSingle. • CKM_IBM_ECDSA_OTHER	CEX6P at level 3.7.11 is required for ReencryptSingle.

		CEX6P at level 3.7.14 is required for CKM_IBM_ECDSA_OTHER.
IBM z15 T01 or z15 T02 - CEX7P	IBM z14 or z14R1 set and • CKM_ECDH1_DERIVE • CKK_IBM_DILITHIUM • CKA_IBM_PROTKEY_EXTRACTABLE • ReencryptSingle. • CKM_IBM_ECDSA_OTHER	CEX7P at level 4.7.10 is required for CKM_ECDH1_DERIVE and CKA_IBM_PROTKEY_EXTRACTABLE CEX7P at level 4.7.10 is required for CKK_IBM_DILITHIUM CEX7P at level 4.7.21 is required for ReencryptSingle. CEX7P at level 4.7.24 is required for CKM_IBM_ECDSA_OTHER

The following table lists the algorithms and uses (by mechanism) that are not allowed when operating in compliance with FIPS 140-2. For more information about how the z/OS PKCS #11 services can be configured to operate in compliance with the FIPS 140-2 standard, see “Operating in compliance with FIPS 140-2”.

Table 87. Restricted algorithms and uses when running in compliance with FIPS 140-2

Algorithm	Mechanisms	Usage disallowed
EC Koblitz	CKM_ECDSA, CKM_ECDH1_DERIVE, CKM_EC_KEY_PAIR_GEN, CKM_ECDSA_SHA1	All

PKCS #11 Coprocessor Access Control Points

The following table lists the Access Control Points that are available on the Enterprise PKCS #11 coprocessors and the PKCS #11 mechanisms or functions that would be disabled for secure keys if the control point is deactivated. A new or a zeroized Enterprise PKCS #11 coprocessor (or domain) comes with an initial set of Access Control Points (ACPs) that are enabled by default. All other ACPs, representing potential future support, are left disabled. When a firmware upgrade is applied to an existing Enterprise PKCS #11 coprocessor , the upgrade might introduce new ACPs. The firmware upgrade does not retroactively enable these ACPs, so they are disabled by default. These ACPs must be enabled with the TKE (or subsequent zeroize) to use the new support they govern.

See the Enabling Access Control Points for PKCS #11 coprocessor firmware section in the Migration topic of the z/OS Cryptographic Services ICSF System Programmer's Guide for the list of default ACPs and those ACPs that need to be enabled with the TKE for PKCS #11 coprocessor firmware upgrades.

The following table lists the Access Control Points that are available on the Enterprise PKCS #11 coprocessors and the PKCS #11 mechanisms or functions that would be disabled for secure keys if the control point is deactivated.

Table 88. PKCS #11 Access Control Points		
Access Control Point name or group	**Mechanism/Function requiring enablement**	**Number**
Cryptographic Algorithms		
Enable support for non-ECDSA/non-EdDSA elliptic curve signature algorithms	EC-SDSA rule for sign and verify, and all subvariants.	67