z/OS

Cryptographic Services
Integrated Cryptographic Service Facility
Cryptographic Services Enhancements for IBM z13s
APAR OA49064

Contents

Chapter 1. Overview

This document describes changes to the Integrated Cryptographic Service Facility (ICSF) product in support of the following:

- Enhancements to ICSF callable services for IBM z13s.
- Support for the new key check value using the CMAC algorithm for the Key Test2 (CSNBKYT2) service.
- Support for AES Galois/Counter mode encryption for the Symmetric Algorithm Encipher (CSNBSAE) and Symmetric Algorithm Decipher (CSNBSAD) services.
- Support for the new key derivation algorithm for the EC Diffie-Hellman (CSNDEDH) service.
- The new Encrypted PIN Translate Enhanced (CSNBPTRE) service to support PAN that is encrypted using format preserving encryption.

These changes are available through the application of the PTF for APAR OA49064 and apply to FMID HCR77B1.

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

- *z/OS Cryptographic Services ICSF Overview*, SC14-7505-04
- *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SC14-7508-04
- *z/OS Cryptographic Services ICSF Administrator's Guide*, SC14-7506-04
- *z/OS Cryptographic Services ICSF System Programmer's Guide*, SC14-7507-04

The technical changes made to the ICSF product by the application of the PTF for APAR OA49064 are indicated in this document by a vertical line to the left of the change.

Chapter 2. Update of z/OS Cryptographic Services ICSF Overview, SC14-7505-04, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF Overview*, SC14-7505-04, 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

In Table 1, letters represent various configurations according to:

- Letter A (**PCIXCC/CEX2C**) - IBM eServer zSeries 990 or IBM eServer zSeries 890 with CP Assist for Cryptographic Functions DES/TDES Enablement and PCIXCC/CEX2C.
- Letter B (**CEX2C/CEX3C**) - z9 EC, z9 BC, z10 EC, and z10 BC with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX2C, or z10 EC and z10 BC with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX3C.
- Letter C (**CEX3C**) - z114/z196 with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX3C.
- Letter D (**CEX3C/CEX4C**) - IBM zEnterprise EC12 and BC12 with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX3 and CEX4C.
- Letter E (**CEX5C**) - IBM z13 and z13s with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX5C.
- Letter F (**CEX4P/CEX5P**) - IBM z13 and z13s or zEnterprise EC12/BC12 with CP Assist for Cryptographic Functions DES/TDES Enablement and CEX4P or CEX5P.
- Letter G (**Open Cryptographic Server**) - IBM zEnterprise EC12 or later system with CP Assist for Cryptographic Functions DES/TDES Enablement and an Open Cryprographic Server.

Table 1. Summary of ICSF callable services support

Service Name	Function	A	B	C	D	E	F	G
Encrypted PIN Translate Enhanced	Reenciphers a PIN block from one PIN-encrypting key to another and changes the PIN block format where the PAN data is encrypted using format preserving encryption.					X		

Chapter 3. Update of z/OS Cryptographic Services ICSF System Programmer's Guide, SC14-7507-04, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF System Programmer's Guide*, SC14-7507-04, for the updates provided by this APAR. Refer to this source document if background information is needed.

Installation, initialization, and customization

Table 2. Exit identifiers and exit invocations

Exit identifiers	Exit invocations
CSFPTRE	Gets control during the Encrypted PIN Translate Enhanced callable service.

Migration

Callable services

The following table summarizes the new and changed callable services for ICSF FMID HCR77B1. For complete reference information on these callable services, refer to *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Table 3. Summary of new and changed ICSF callable services

Callable service	FMID	Description
ECC Diffie-Hellman	HCR77B1	**Changed:** Support for new derivation algorithm.
Encrypted PIN Translate Enhanced	HCR77B1	**New:** Reformat a PIN block where the PAN data is encrypted Visa Data Secure Platform (Visa DSP) processing.
Key Test2	HCR77B1	**Changed:** Support new key check value algorithm based on CMAC for DES and AES.
PKA Key Token Build	HCR77B1	**Changed:** Support for key derivation section for EC private keys added.
Symmetric Key Decipher	HCR77B1	**Changed:** Support Galois/Counter Mode for AES.
Symmetric Key Decipher	HCR77B1	**Changed:** Support Galois/Counter Mode for AES.

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:

HCR77B1
CSFPTRE

Installation exits

Table 4. Services and their ICSF names

Service	*ICSF name*
Encrypted PIN Translate Enhanced	**CSFPTRE**

Diagnosis reference information

RMF measurements table

Table 5 describes the contents of the performance measurements for RMF. The count fields are double-word length.

Table 5. RMF measurements record format

Offset (Dec)	**Number of bytes**	**Field name**	**Description**
240	8	DACC_ENT_ID	Identifier of count array - character PTR. The Encrypted PIN Translate and Encrypted PIN Translate Enhanced services will collect data as follows: • Collect the number of service calls only.
248	8	DACC_ENT_SVC_CNT	Count of PTR and PTRE service calls.

Chapter 4. Update of z/OS Cryptographic Services ICSF Application Programmer's Guide, SC14-7508-04, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SC14-7508-04, for the updates provided by this APAR. Refer to this source document if background information is needed.

Introducing symmetric key cryptography and using symmetric key callable services

DES key types

The DES keys are 64-bit, 128-bit, and 192-bit keys that use the DES algorithm to perform the cryptographic function. A 64-bit key is referred to as a single-length key. A 128-bit key is referred to as a double-length key. Triple-length keys are 192-bits in length. Only DATA keys can be triple-length.

For installations that do not support double-length key-encrypting keys, effective single-length keys are provided. For an effective single-length key, the clear key value of the left key half equals the clear key value of the right key half.

Table 6. Descriptions of DES key types and service usage

DES key type	Usable with services
DATA class (data operation keys) These key are used to encrypt and decrypt data. Single-length keys can be used to generate and verify MACs and CVVs. DATA keys can be single-length, double-length, or triple-length. DATAM and DATAMV keys are double-length.	
DATA	Authentication Parameter Generate, Cipher Text Translate2, CVV Key Combine, Decipher, Encipher, EMV Verification Functions, Field Level Decipher, Field Level Encipher, MAC Generate, MAC Verify, Symmetric Key Encipher, Symmetric Key Decipher, VISA CVV Generate, VISA CVV Verify
DATAM	MAC Generate, MAC Verify
DATAMV	MAC Verify
Cipher class (data operation keys) These key are used to encrypt and decrypt data. The keys can be single-length or double-length.	
CIPHER	Cipher Text Translate2, Decipher, Encipher, Encrypted PIN Translate Enhanced, FPE Decipher, FPE Encipher, FPE Translate
DECIPHER	Cipher Text Translate2, Decipher, Encrypted PIN Translate Enhanced, FPE Encipher, FPE Translate
ENCIPHER	Cipher Text Translate2, Encipher, FPE Decipher, FPE Translate
CIPHERXL class (cipher text translate keys) These key are used to translate cipher text. The keys are double--length.	
CIPHERXI	Cipher Text Translate2 (translate inbound key only)
CIPHERXL	Cipher Text Translate2 (translate inbound and outbound key)
CIPHERXO	Cipher Text Translate2 (translate outbound key only)

Table 6. Descriptions of DES key types and service usage (continued)

DES key type	Usable with services
MAC class (data operation keys) These keys are used to generate and verify MACs, CVVs, and CSCs. The keys can be single-length or double-length keys.	
MAC	CVV Key Combine, MAC Generate, MAC Verify, Transaction Validation, VISA CVV Generate, VISA CVV Verify
MACVER	CVV Key Combine, MAC Verify, Transaction Validation, VISA CVV Verify
PIN class These keys are used generate and verify PINs and PIN offsets. The keys are double-length keys.	
PINGEN	Clear PIN Generate, Clear PIN Generate Alternate, Encrypted PIN Generate, Recover PIN from Offset
PINVER	Encrypted PIN Verify
These keys are used wrap and unwrap PIN blocks:	
IPINENC	Authentication Parameter Generate, Clear PIN Generate Alternate, EMV Scripting Service, Encrypted PIN Translate, Encrypted PIN Translate Enhanced, Encrypted PIN Verify, PIN Change/Unblock, Secure Messaging for PINs
OPINENC	Clear PIN Encrypt, Clear PIN Generate Alternate, EMV Scripting Service, Encrypted PIN Generate, Encrypted PIN Translate, Encrypted PIN Translate Enhanced, PIN Change/Unblock, Recover PIN from Offset
Key-encrypting key class These keys are used to wrap other keys. The keys are double-length keys.	
EXPORTER	Control Vector Translate, Data Key Export, Derive ICC MK, ECC Diffie-Hellman, Generate Issuer MK, Key Export, Key Generate, Key Test2, Key Test Extended, Key Translate, Key Translate2, PKA Key Generate, PKA Key Translate, Prohibit Export Extended, Remote Key Export, Secure Messaging for Keys, Symmetric Key Generate, TR-31 Export, TR-31 Import, Unique Key Derive
IMPORTER	Control Vector Translate, Data Key Import, ECC Diffie-Hellman, Generate Issuer MK, Key Generate, Key Import, Key Test2, Key Test Extended, Key Translate, Key Translate2, Multiple Secure Key Import, PKA Key Generate, PKA Key Import, PKA Key Translate, Prohibit Export Extended, Remote Key Export, Restrict Key Attribute, Secure Key Import, Secure Messaging for Keys, Symmetric Key Generate, TR-31 Export, TR-31 Import
IMP-PKA	PKA Key Import, Remote Key Export, Trusted Block Create
IKEYXLAT, OKEYXLAT	Control Vector Translate, Key Translate, Key Translate2, TR-31 Export,TR-31 Import
Key-generate key class These keys are used to derive keys. The keys are double-length keys. The key usage flags in the control vector determine which services the KEYGENKY key may be used with.	
KEYGENKY	Diversified Key Generate, Encrypted PIN Translate, Encrypted PIN Translate Enhanced, Encrypted PIN Verify, FPE Decipher, FPE Encipher, FPE Translate, Unique Key Derive

Table 6. Descriptions of DES key types and service usage (continued)

DES key type	Usable with services
DKYGENKY	Derive ICC MK, Derive Session Key, Diversified Key Generate, EMV Scripting Service, EMV Transaction (ARQC/ARPC) Service, EMV Verification Functions, Generate Issuer MK, PIN Change/Unblock
Cryptographic-variable class These keys are used in the special verbs that operate with cryptographic variables The keys are single-length keys.	
CVARENC	Cryptographic Variable Encipher
CVARXCVL	Control Vector Translate
CVARXCVR	Control Vector Translate
Secure-messaging class (data operation keys) These keys are used to encrypt keys or PINs. The keys are double-length keys. The key usage flags in the control vector determine which services the key may be used with.	
SECMSG	Diversified Key Generate, Secure Messaging for Keys, Secure Messaging for PINs

Encrypted PIN Translate Enhanced Callable Service (CSNBPTRE and CSNEPTRE)

To reformat a PIN block where the PAN data is encrypted using format preserving encryption, call the Encrypted PIN translation enhanced callable service. You must identify the input PIN-encrypting key that originally enciphers the PIN, the output PIN-encrypting key that you want the callable service to use to encipher the PIN, and the key to decipher the PAN. All of these keys may be derived using a key-generating key. The PAN data cannot be changed when reformatting the PIN block. See "Encrypted PIN Translate Enhanced (CSNBPTRE and CSNEPTRE)" on page 39 for more information.

Summary of callable services

Table 7 lists the callable services described in this publication, and their corresponding verbs. The figure also references the topic that describes the callable service.

Table 7. Summary of ICSF callable services

Service	Service name	Function
Chapter 8, "Financial Services,"		
CSNBPTRE CSNEPTRE	Encrypted PIN Translate Enhanced	Reformat a PIN block where the PAN data is encrypted using format preserving encryption. Unique key per transaction key derivation is supported.

Managing Symmetric Cryptographic Keys

ECC Diffie-Hellman (CSNDEDH and CSNFEDH)

Use the ECC Diffie-Hellman callable service to create:

- Symmetric key material from a pair of ECC keys using the Elliptic Curve Diffie-Hellman protocol and the static unified model key agreement scheme.
- “Z” – The “secret” material output from D-H process.

Output may be one of the following forms:
- Internal CCA Token (DES or AES): AES keys are in the "Variable-length Symmetric Key Token" format. DES keys are in the "DES Internal Key Token" format.
- External CCA Token (DES or AES): AES keys are in the "Variable-length Symmetric Key Token" format. DES keys are in the "DES External Key Token" format.
- “Z” – The “secret” material output from D-H process.

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

Format

```
CALL CSNDEDH(
            return_code,
            reason_code,
            exit_data_length,
            exit_data,
            rule_array_count,
            rule_array,
            private_key_identifier_length,
            private_key_identifier,
            private_KEK_key_identifier_length,
            private_KEK_key_identifier,
            public_key_identifier_length,
            public_key_identifier,
            chaining_vector_length,
            chaining_vector,
            party_identifier_length,
            party_identifier,
            key_bit_length,
            reserved_length,
            reserved,
            reserved2_length,
            reserved2,
            reserved3_length,
            reserved3,
            reserved4_length,
            reserved4,
            reserved5_length,
            reserved5,
            output_KEK_key_identifier_length,
            output_KEK_key_identifier,
            output_key_identifier_length,
            output_key_identifier)
```

Parameters

return_code

Direction	**Type**
Output	Integer

The return code specifies the general result of the callable service.

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.

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. Valid
| values are 1, 2, 3, 4, 5, and 6.

rule_array

Direction	Type
Input	String

The *rule_array* parameter is an array of keywords. 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. The rule_array keywords are:

Table 8. Keywords for ECC Diffie-Hellman

Keyword	Meaning
Key agreement (one required)	
DERIV01	Use input skeleton key-token and derive one element of any key pair. Denotes ANS X9.63 protocol static unified model key-agreement scheme (see NIST SP800-56A). Initiator and responder must have a sufficient level of trust such that they each derive only one element of any key pair. The DERIV01 rule is designed for CCA to CCA interaction.
DERIV02	Use input skeleton key-token and derive one element of any key pair. Denotes key derivation function ANSI-X9.63-KDF (see Section 5.6.3 of ANSI X9.63-2011). Initiator and responder must have a sufficient level of trust such that they each derive only one element of any key pair.
PASSTHRU	Skip Key derivation step and return raw “Z" material.

Table 8. Keywords for ECC Diffie-Hellman (continued)

Keyword	Meaning
Transport Key Type (one optional if output KEK key identifier is present)	
OKEK-DES	The output KEK key identifier is a "DES" KEK token.
OKEK-AES	The output KEK key identifier is a "AES" KEK token.
Output Key Type (one optional if output key identifier is present)	
KEY-DES	The output key identifier is a "DES" skeleton token.
KEY-AES	The output key identifier is an "AES" skeleton token.
Hash type (one optional, only valid with DERIV02)	
SHA-224	Specifies the use of the SHA-224 method.
SHA-256	Specifies the use of the SHA-256 method. This is the default.
SHA-384	Specifies the use of the SHA-384 method.
SHA-512	Specifies the use of the SHA-512 method.
Key Wrapping Method (one optional, only supported when the output type is DES)	
USECONFG	Specifies that the configuration setting for the default wrapping method is to be used to wrap the key. This is the default.
WRAP-ENH	Specifies that the new enhanced wrapping method is to be used to wrap the key.
WRAP-ECB	Specifies that the original wrapping method is to be used.
Translation Control (one optional, only supported when the output type is DES)	
ENH-ONLY	Specify this keyword to indicate that the key once wrapped with the enhanced method cannot be wrapped with the original method. This restricts translation to the original method. If the keyword is not specified translation to the original method will be allowed. This turns on bit 56 (ENH ONLY) in the control vector. This keyword is not valid if processing a zero CV data key.

private_key_identifier_length

Direction	Type
Input	Integer

The length of the *private_key_identifier* parameter in bytes. 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 3500.

private_key_identifier

Direction	Type
Input	String

The *private_key_identifier* must contain an internal or an external token or a label of an internal or external ECC key. The ECC key token must contain a public-private key pair. When the key agreement keyword is DERIV01, a clear key will be accepted.

The ECC curve type and size must be the same as the type (Prime or Brainpool) and size of the ECC key-token specified by the public key identifier parameter. The key-usage flag byte (offset 50 in the private-key section) of the

ECC key-token identified by the private key identifier parameter must permit key establishment (either KEY-MGMT or KM-ONLY).

For keyword DERIV02, the key identifier must contain a key-derivation section, type X'23' (see key-derivation in Section 4.2 and Table 14 of ANSI X9.63-2011).

private_KEK_key_identifier_length

Direction	**Type**
Input	Integer

The length of the *private_KEK_key_identifier* in bytes. If the *private_KEK_key_identifier* contains a label, the value must be 64. Otherwise, the value must be between the actual length of the token and 900. If the *private_key_identifier* contains an internal ECC token, this value must be a zero.

private_KEK_key_identifier

Direction	**Type**
Input	String

The key-encrypting key that the ECC private key token is encrypted under. This applies when the *private_key_identifier* is an external ECC token. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm must be AES and the key must be either EXPORTER or IMPORTER.

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

public_key_identifier_length

Direction	**Type**
Input	Integer

The length of the *public_key_identifier* in bytes. 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 3500.

public_key_identifier

Direction	**Type**
Input	String

The *public_key_identifier* parameter must contain an ECC public token or the label of an ECC Public token. The *public_key_identifier* specifies the other party's ECC public key which is enabled for key management functions. If the *public_key_identifier* identifies a token containing a public-private key pair, no attempt to decrypt the private part will be made.

chaining_vector_length

Direction	**Type**
Input/Output	Integer

The *chaining_vector_length* parameter must be zero.

chaining_vector

Direction	Type
Input/Output	String

The *chaining_vector* parameter is ignored.

party_identifier_length

Direction	Type
Input/Output	Integer

| The length of the *party_identifier* parameter in bytes. For the DERIV01 keyword,
| the value must be between 8 and 64, inclusive. For the DERIV02 keyword, the
| value must be between 0 and 256, inclusive. When the PASSTHRU rule array
| keyword is specified, the value must be 0 and the *party_identifier* parameter is
| ignored.

party_identifier

Direction	Type
Input/Output	String

The *party_identifier* parameter contains the entity identifier information. This information should contain the both entities data according to NIST SP800-56A
| Section 5.8 when the DERIV01 rule array keyword is specified. For DERIV02,
| this information should contain the optional shared data according to Section
| 5.6.3 of ANSI X9.63-2011.

key_bit_length

Direction	Type
Input/Output	Integer

The key bit length parameter contains the number of bits of key material to derive and place in the provided key token. The value must be 0 if the PASSTHRU rule array keyword was specified. Otherwise it must be 64 - 2048.

reserved_length

Direction	Type
Input/Output	Integer

The *reserved_length* parameter must be zero.

reserved

Direction	Type
Input/Output	String

This parameter is ignored.

reserved2_length

Direction	Type
Input/Output	Integer

The *reserved2_length* parameter must be zero.

reserved2

Direction	Type
Input/Output	String

This parameter is ignored.

reserved3_length

Direction	Type
Input/Output	Integer

The *reserved3_length* parameter must be zero.

reserved3

Direction	Type
Input/Output	String

This parameter is ignored.

reserved4_length

Direction	Type
Input/Output	Integer

The *reserved4_length* parameter must be zero.

reserved4

Direction	Type
Input/Output	String

This parameter is ignored.

reserved5_length

Direction	Type
Input/Output	Integer

The *reserved5_length* parameter must be zero.

reserved5

Direction	Type
Input/Output	String

This parameter is ignored.

output_KEK_key_identifier_length

Direction	Type
Input	Integer

| The length of the *output_KEK_key_identifier* parameter in bytes. If the
| *output_KEK_key_identifier* contains a label, the value must be 64. Otherwise, the
| value must be between the actual length of the token and 900. The *output_KEK_key_identifier_length* must be zero if *output_key_identifier* will contain an internal token or if the PASSTHRU rule array keyword was specified.

output_KEK_key_identifier

Direction	Type
Input/Output	String

The *output_KEK_key_identifier* contains a KEK key token or the label of a KEK key if the *output_key_identifier* will contain an external ECC token. Otherwise this field is ignored.

If the output KEK key identifier identifies a DES KEK, then it must be an IMPORTER or an EXPORTER key type, and have the export bit set. The XLATE bit is not checked. If the output KEK key identifier identifies an AES KEK, then it must be either an IMPORTER or an EXPORTER key type and have the export/import bit set in key usage field 1 and the derivation bit set in key usage field 4.

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

output_key_identifier_length

Direction	Type
Input/Output	Integer

| The length of the *output_key_identifier* parameter in bytes. The service checks the field to ensure it is at least equal to the size of the token to return. On return from this service, this field is updated with the exact length of the key token created. The maximum allowed value is 900 bytes.

output_key_identifier

Direction	Type
Input/Output	String

| On input, the *output_key_identifier* must contain a skeleton token (DERIV01 or
| DERIVE02) or a null token (PASSTHRU).

On output, the *output_key_identifier* will contain:

- An internal or an external key token containing the generated symmetric key material.
- "Z" data (in the clear) if the PASSTHRU rule array keyword was specified.

If this variable specifies an external DES key token then the output KEK key identifier must identify a DES KEK key token. If this specifies an external key token other than a DES key token then the output KEK key identifier must identify an AES KEK key token.

Restrictions

The NIST security strength requirements will be enforced, with respect to ECC Curve type (input) and derived key length.

Only the following key types will be generated, skeleton key tokens of any other type will fail.

- DES: (Legacy DES token)
  - CIPHER
  - CIPHERXI
  - CIPHERXL
  - CIPHERXO
  - DECIPHER

  - ENCIPHER
  - IMPORTER
  - EXPORTER
  - IMP-PKA
- AES
  - DATA (Legacy AES token)
  - CIPHER (Variable-length symmetric key-token)
  - IMPORTER (Variable-length symmetric key-token)
  - EXPORTER (Variable-length symmetric key-token)

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.

This table lists the valid key bit lengths and the minimum curve size required for each of the supported output key types.

Table 9. Valid key bit lengths and minimum curve size required for the supported output key types.

Output Key ID type	Valid Key Bit Lengths	Minimum Curve Required
DES	64	P160
	128	P160
AES	128	P256
	192	P384
	256	P512

If the output key-encrypting key identifier is a weaker key than the key being generated, 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.

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 or the key-encrypting key is a double-length key.

Access control points

The ECC Diffie-Hellman callable service requires the **ECC Diffie-Hellman Callable Service** access control point to be enabled in the domain role.

Specifying the PASSTHRU rule array keyword requires that the **ECC Diffie-Hellman – Allow PASSTHRU** access control point be enabled in the domain role.

Specifying the DERIV02 rule array keyword requires that the **ECC Diffie-Hellman – Allow DERIV02** access control point be enabled in the domain role.

If the *output_key_identifier* parameter references a DES key token and the wrapping method specified in not the default method, then the **ECC Diffie-Hellman – Allow key wrap override** access control point must be enabled in the domain role.

Each Elliptic Curve type supported has its own access control point. The access control point must be enabled to use the curve type and strength.
- ECC Diffie-Hellman – Allow Prime Curve 192
- ECC Diffie-Hellman – Allow Prime Curve 224
- ECC Diffie-Hellman – Allow Prime Curve 256
- ECC Diffie-Hellman – Allow Prime Curve 384
- ECC Diffie-Hellman – Allow Prime Curve 521
- ECC Diffie-Hellman – Allow BP Curve 160
- ECC Diffie-Hellman – Allow BP Curve 192
- ECC Diffie-Hellman – Allow BP Curve 224
- ECC Diffie-Hellman – Allow BP Curve 256
- ECC Diffie-Hellman – Allow BP Curve 320
- ECC Diffie-Hellman – Allow BP Curve 384
- ECC Diffie-Hellman – Allow BP Curve 512

To prevent a weaker key from being used to generate a stronger key, enable the **ECC Diffie-Hellman – Prohibit weak key generate** access control point in the domain role.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service.

Table 10. ECC Diffie-Hellman required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890		This callable service is not supported.
IBM System z9 EC IBM System z9 BC		This callable service is not supported.
IBM System z10 EC IBM System z10 BC		This callable service is not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	ECC Clear Key and Internal tokens support requires the Sep. 2010 licensed internal code (LIC). ECC External and Diffie-Hellman support requires Sep. 2011 licensed internal code (LIC). The DERIV02, SHA-224, SHA-256, SHA-384, and SHA-512 keywords are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	The DERIV02, SHA-224, SHA-256, SHA-384, and SHA-512 keywords are not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	The DERIV02, SHA-224, SHA-256, SHA-384, and SHA-512 keywords requires the March 2016 or later licensed internal code (LIC).

Key Test2 (CSNBKYT2 and CSNEKYT2)

Use this callable service to generate or verify a secure, cryptographic verification pattern (also referred to as a key check value) for AES, DES and HMAC keys. The key to test can be in the clear, encrypted under the master key, or encrypted under a key-encrypting key. Keywords in the *rule_array* specify whether the callable service generates or verifies a verification pattern.

For AES, key tokens may be either internal, fixed-length (version 04) tokens, or external and internal variable-length (version 05) tokens.

For DES, key tokens may be external and internal, fixed-length (versions 00 or 01) tokens, external TR-31 key blocks, or external variable-length tokens with a DESUSECV key.

For HMAC, key tokens are external and internal variable-length (version 05) key tokens.

The callable service name for AMODE(64) invocation is CSNEKYT2.

Format

```
CALL CSNBKYT2(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             key_encrypting_key_identifier_length,
             key_encrypting_key_identifier,
             reserved_length,
             reserved,
             verification_pattern_length,
             verification_pattern )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service.

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.

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, 3, 4, or 5.

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 11. Keywords for Key Test2 Control Information

Keyword	Meaning
Token algorithm (Required)	
AES	Specifies the key token is an AES key token.
DES	Specifies the key token is a DES token. CCA internal, CCA external, and TR-31 token types are supported. Clear keys are not supported for this rule.
HMAC	Specifies the key token is an HMAC key token.
Process rule (One required)	
GENERATE	Generate a verification pattern for the specified key.
VERIFY	Verify that a verification pattern matches the specified key.
Verification pattern calculation algorithm (Optional)	
CMACZERO	Specifies that the verification pattern for AES and DES keys will be calculated by computing a MAC upon a data block of 0x00 bytes using the CMAC algorithm.
ENC-ZERO	Specifies that the verification pattern for AES and DES keys will be calculated by encrypting a data block filled with 0x00 bytes using the ECB mode. This is the default method for DES.

Table 11. Keywords for Key Test2 Control Information (continued)

Keyword	Meaning
SHA-256	Specifies that the verification pattern will be calculated for an AES token using the same method as the Key Test service with the SHA-256 rule. This the default method for AES. This rule can be used to verify that the same key value is present in a version 4 DATA token and version 5 AES CIPHER token or to verify that the same key value is present in a version 5 AES complementary key pairs.
SHA2VP1	Specifies that the verification pattern will be calculated using the SHA-256 algorithm. For more information, see 'SHAVP1 Algorithm'. This is the default and only method available for HMAC.
Token type rule (One required if the DES token algorithm is specified and a TR-31 token or DESUSECV token is passed; not valid otherwise)	
TR-31	Specifies that *key_identifier* contains a TR-31 key block.
AESKWCV	Specifies that *key_identifier* contains an external variable length symmetric key token whose type is DESUSECV. The IKEK-AES keyword must be specified for the KEK identifier rule.
KEK identifier rules (One required if the AESKWCV token type is specified)	
IKEK-AES	The wrapping KEK for the key to test is an AES KEK. This is the default for AES and HMAC Token algorithms.
IKEK-DES	The wrapping KEK for the key to test is a DES KEK. This is the default for DES Token algorithm, and is only allowed with the DES Token algorithm.
IKEK-PKA	The wrapping KEK for the key to test is an RSA or (other key stored in PKA key storage.) This is not the default for any Token algorithm and must be specified if an RSA KEK is used. This rule is not allowed with DES Token algorithm.

key_identifier_length

Direction	Type
Input	Integer

The length of the key_identifier in bytes. The maximum value is 9992.

key_identifier

Direction	Type
Input/Output	String

The key for which to generate or verify the verification pattern. This is an internal or external token or the 64-byte label of a key in the CKDS. This token may be a DES internal or external token, AES internal version ‘04’X token, internal or external variable-length symmetric token, or a TR-31 key block.

Clear DES tokens are not supported.

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

key_encrypting_key_identifier_length

Direction	Type
Input	Integer

The length of the *key_encrypting_key_identifier* parameter. When *key_identifier* is an internal token, the value must be zero.

If *key_encrypting_key_identifier* is a label for either the CKDS (IKEK-AES or IKEK-DES rules) or PKDS (IKEK-PKA rule), the value must be 64. If *key_encrypting_key_identifier* is an AES KEK, the value must be between the actual length of the token and 725. If *key_encrypting_key_identifier* is a DES KEK, the value must be 64. If *key_encrypting_key_identifier* is an RSA KEK, the maximum length is 3500.

key_encrypting_key_identifier

Direction	Type
Input/Output	String

When *key_encrypting_key_identifier_length* is non-zero, *key_encrypting_key_identifier* contains an internal key token containing the key-encrypting key, or a key label.

If the key identifier supplied was an AES or DES token encrypted under the old master key, the token will be returned encrypted under the current master key.

reserved_length

Direction	Type
Input	Integer

The length of the reserved parameter. The value must be zero.

reserved

Direction	Type
Input/Output	String

This parameter is ignored.

verification_pattern_length

Direction	Type
Input/Output	Integer

The length of the *verification_pattern* parameter in bytes.

Table 12. Length of the verification pattern for each algorithm supported

Calculation algorithm	Length of the verification pattern in bytes
CMACZERO	AES: 5 DES: 3
ENC-ZERO	8
SHA-256	8
SHA2VP1	9

On input for GENERATE, the value should be the size of the buffer for the *verification_pattern* parameter.

On input for VERIFY, the length must be the length of the verification pattern supplied in the *verification_pattern* parameter.

On output for GENERATE, the parameter will be updated for the length of the *verification_pattern* returned.

verification_pattern

Direction	Type
Input/Output	String

For GENERATE, the verification pattern generated for the key.

For VERIFY, the supplied verification pattern to be verified.

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.

You can generate the verification pattern for a key when you generate the key. You can distribute the pattern with the key and it can be verified at the receiving node. In this way, users can ensure using the same key at the sending and receiving locations. You can generate and verify keys of any combination of key forms: clear, operational or external.

Access control point

The access control point in the domain role that controls the function of this service is **Key Test and Key Test 2**. This access control point cannot be disabled. It is required for ICSF master key validation.

Some of the verification pattern calculation algorithm keywords require an additional access control point to be enabled. Keywords not in the table are always available.

Table 13. Required access control points for Key Test2

Rule array keyword	Key algorithm	Access control point
CMACZERO	AES	Key Test2 - AES, CMACZERO
CMACZERO	DES	Key Test2 - DES, CMACZERO
ENC-ZERO	AES	Key Test2 - AES, CMACZERO

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service.

Table 14. Key Test2 required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890		This service is not supported.
IBM System z9 EC IBM System z9 BC		This service is not supported.

Table 14. Key Test2 required hardware (continued)

Server	Required cryptographic hardware	Restrictions
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 support requires the November 2013 or later licensed internal code (LIC). DES/AES key support requires 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 CMACZERO keywords are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	DK AES PIN support requires the September 2013 or later licensed internal code (LIC). The AESKWCV keyword requires the September 2013 or later licensed internal code (LIC). The CMACZERO keyword is not supported.
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	The CMACZERO keyword requires the March 2016 or later licensed internal code (LIC).

Protecting Data

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 unpaded clear text data. PKCS padding is described in 'PKCS Padding Method'.

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.

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.

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 15. 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.

Table 15. Symmetric Algorithm Decipher Rule Array Keywords (continued)

Keyword	Meaning
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.
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 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	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
Ignored	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 and PKCS-PAD, 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 2^{32}-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, and PKCS-PAD 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/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, 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, and PKCS-PAD, 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.

cipher_text

Direction	Type
Input	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 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
Ignored	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.

Table 16. Symmetric Algorithm Decipher required hardware

Server	**Required cryptographic hardware**	**Restrictions**
IBM eServer zSeries 990 IBM eServer zSeries 890		This service is not supported.
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.
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.
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.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Keywords GCM and ONLY 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).

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 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.

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.

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

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 17. 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).
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	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'. 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	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. 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 ENCRYPT 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. If there is an error in processing, this value will be set to zero on output. Otherwise, it will be unchanged.

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

key_parms

Direction	Type
Ignored	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.

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 and PKCS-PAD, 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 2^{32}-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, and PKCS-PAD 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.

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, the value must be at least 32. For ECB and GCM, this 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 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 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.

clear_text

Direction	Type
Input	String

The text to be enciphered.

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 is used, 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.

cipher_text

Direction	Type
Output	String

The enciphered 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
Ignored	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 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 **Symmetric Algorithm Encipher - secure AES keys** access control point
| controls the function of this service. Use of the GCM processing rule requires that
| the **Symmetric Algorithm Encipher – 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.

Table 18. Symmetric Algorithm Encipher required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890		This service is not supported.
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.
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.
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.

Table 18. Symmetric Algorithm Encipher required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	Keywords GCM and ONLY 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).

Financial Services

Format preserving encryption

Format preserving encryption (FPE) is a method of encryption where the resulting cipher text has the same form as the input clear text. The form of the text can vary according to use and the application. One example is a 16 digit credit card number. After using FPE to encrypt a credit card number, the resulting cipher text is another 16 digit number. In this example of the credit card number, the output cipher text is limited to numeric digits only.

The FPE services require some knowledge of the input clear text character set in order to create the appropriate output ciphertext. The CSNBFPEE, CSNBFPED, CSNBFPET, and CSNBPTRE callable services use the following tables to determine valid character sets for the clear text input parameters:

Encrypted PIN Translate Enhanced (CSNBPTRE and CSNEPTRE)

Use the Encrypted PIN Translate Enhanced callable service to change the format of a PIN block where the PAN field is enciphered using format preserving encryption. The service supports translation of PIN blocks whose PAN information has been enciphered using the Visa Merchant Data Secure (VMDS) standard and Visa Format Preserving Encryption (VFPE) encryption methods. Change of PAN data is not allowed.

PIN blocks are sometimes formatted using the PAN information. For this service, either the input PIN block profile or the output PIN block profile must specify a PIN block format that incorporates a PAN. The PIN block formats which incorporate a PAN are ISO-0, ISO-3, and Visa Format 4. Change of PAN data is not allowed.

Unique-key-per-transaction key derivation support is available for the *input_PIN_encrypting_key_identifier*, *output_PIN_encrypting_key_identifier*, and the *PAN_key_identifier* parameters. Optional rule array keywords determines which keys are to be derived and which key identifier parameters contains the key-generating key.

VMDS enciphered PAN data can be enciphered using DUKPT key management or static TDES key management. The enciphered PAN could be enciphered with the CBC or VFPE mode. The VMDS standard requires that the same key management scheme and type of keys be used for both the PIN and the PAN.

For VMDS, the following pairings are supported:

Table 19. VMDS pairings for enciphered PAN data

	Source		Target	
Function	**Key management**	**VMDS option**	**Key management**	**VMDS option**
Translation	DUKPT	Standard CBC	Static TDES non-DUKPT	Standard CBC
		VFPE		
	Static TDES non-DUKPT	Standard CBC		

The VMDS standard refers to double length, non-DUKPT keys as Zone Encryption keys.

To use this service, specify the following information:

- The mode of operation with a keyword in the rule array: **REFORMAT**.
- Optionally, the method of PIN extraction with a rule-array keyword.
- The Input and Output PIN block encrypting keys, or the key-encrypting keys used to derive the PIN block enciphering keys (rule array keywords **DUKPT-IP**, **DUKPT-OP**, or **DUKPT-BH**).
- The PAN-encrypting key or base key used to derive the PAN-encrypting key (rule array keywords **IN-DUKPT**, **OUTDUKPT**, or **STATIC**).
- The Input PIN block.
- The Input and Output PIN profiles. For UKPT processing, the profiles are extended to 48 bytes with a 24-byte current-key serial number (CKSN) extension.
- The Input PAN data as required by the selected PIN-block formats.
- An output PIN-block sequence number. Specify a value of 99999.
- For VMDS processing, you must also specify:
  - Processing algorithm: **VMDS**.
  - PAN input character set: **PAN8BITA** or **PAN4BITX**.
  - PAN input data encryption algorithm: **TDES**.
  - PAN input data mode: **CBC** or **VFPE**.
  - If using VFPE mode encryption, check digit compliance indicator.
  - If using CBC mode encryption, the data decryption key is needed to recover the enciphered PAN.

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

Format

```
CALL CSNBPTRE(
            return_code,
            reason_code,
            exit_data_length,
            exit_data,
            rule_array_count,
            rule_array,
            input_PIN_key_identifier_length,
            input_PIN_key_identifier,
            output_PIN_key_identifier_length,
            output_PIN_key_identifier,
            PAN_key_identifier_length,
            PAN_key_identifier,
            input_PIN_profile_length,
            input_PIN_profile,
            PAN_data_length,
PAN_data,
input_PIN_block_length,
input_PIN_block,
output_PIN_profile_length,
output_PIN_profile,
sequence_number,
output_PIN_block_length,
output_PIN_block,
reserved1_length,
reserved1,
reserved2_length,
reserved2 )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service.

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.

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 between 6 and 9 inclusive.

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 Encrypted PIN Translate Enhanced

Keyword	Meaning
Mode (required)	
REFORMAT	Specifies that either or both the PIN-block format and the PIN-block encryption are to be changed. If the PIN-extraction method is not chosen by default, another element in the rule array must specify one of the keywords that indicates a PIN-extraction method.
Processing method (required)	
VMDS	Specifies the Visa Merchant Data Secure method be used for processing.
Input PAN data key management method (one required) These keywords are used to define the PAN-encrypting key used to decrypt the *PAN_data* parameter.	
IN-DUKPT	Specifies that the key to be used to decrypt the PAN data is to be derived using the key specified in the *input_PIN_key_identifier*. See the description of the *input_PIN_key_identifier* for the requirements of the key. The DUKPT-BH or DUKPT-IP keyword is required.
OUTDUKPT	Specifies that the key to be used to decrypt the PAN data is to be derived using the key specified in the *output_PIN_key_identifier*. See the description of the *output_PIN_key_identifier* for the requirements of the key. The DUKPT-BH or DUKPT-OP keyword is required.
STATIC	Specifies that key is supplied in the *PAN_key_identifier* parameter is to be used to decrypt the PAN data.
Input data algorithm (one required)	
TDES	Specifies Triple-DES encryption was used for the PAN data.
Input data mode (one required)	
CBC	Specifies CBC mode encryption was used for the PAN data. This is the mode for the Standard Encryption option.
VFPE	Specifies Visa format preserving mode encryption was used for the PAN data.
PAN input character set (one required)	
PAN4BITX	Specifies the PAN data character set is 4-bit hexadecimal. Two digits per byte. Not valid with the CBC rule.
PAN8BITA	Specifies the PAN data character set is normal ASCII represented in binary format. Not valid with CBC rule.
PAN-EBLK	Specifies the PAN data is in a CBC encrypted block. Valid only with CBC rule.
PAN check digit compliance (one required if mode VFPE and PAN input character set keyword present; otherwise, not allowed)	
CMPCKDGT	Last digit of the PAN data contains a compliant check digit per ISO/IEC 7812-1.
NONCKDGT	Last digit of the PAN data contains does not contain a compliant check digit per ISO/IEC 7812-1.

Table 20. Rule array keywords for Encrypted PIN Translate Enhanced (continued)

Keyword	Meaning
Unique key per transaction (one optional) These keywords are for the PIN-encrypting keys.	
DUKPT-BH	Specifies that the input and output PIN-encrypting keys are to be derived using the key-generating key specified in the respective parameters. See the descriptions of the *input_PIN_key_identifier* and *output_PIN_key_identifier* parameters for the requirements of the keys.
DUKPT-IP	Specifies that the input PIN-encrypting key is to be derived using the key-generating key specified in the *input_PIN_key_identifier* parameter. See the description of the *input_PIN_key_identifier* for the requirements of the key.
DUKPT-OP	Specifies that the output PIN-encrypting key is to be derived using the key-generating key specified in the *output_PIN_key_identifier* parameter. See the description of the *output_PIN_key_identifier* for the requirements of the key.
PIN-extraction method (one, optional)	See 'PIN Block Format and PIN Extraction Method Keywords' 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 method listed in 'PIN Block Format and PIN Extraction Method Keywords' for the PIN block format will be the default method.

`input_PIN_key_identifier_length`

Direction	Type
Input	Integer

Specifies the length of the *input_PIN_key_identifier* parameter in bytes. The value must be 64.

`input_PIN_key_identifier`

Direction	Type
Input/Output	String

The identifier of the key to decrypt the input PIN block or the key-generating key to be used to derive the key to decrypt the input PIN block. The key-generating key can optionally be used to derive the key to decrypt the PAN data. The key identifier is an operational token or the key label of an operational token in key storage.

If you do not use the UKPT process or you specify the DUKPT-OP rule array keyword, the key token must contain the PIN-encrypting key to be used to decipher the input PIN block. The key algorithm must be DES, the key type must be IPINENC, and the key usage REFORMAT bit must be enabled.

If you use the UKPT process for the input PIN block by specifying the DUKPT-IP or DUKPT-BH rule array keyword, the key token must contain the key-generating key to derive the PIN-encrypting key. If you have also specified the IN-DUKPT keyword, the key will be used to derive the key to decrypt the PAN data. The key algorithm must be DES, the key type must be KEYGENKY, and the key usage UKPT bit 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.

output_PIN_key_identifier_length

Direction	Type
Input/Output	Integer

Specifies the length of the *output_PIN_key_identifier* parameter in bytes. The value must be 64.

output_PIN_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to encrypt the output PIN block or the key-generating key to be used to derive the key to encrypt the output PIN block. The key-generating key can optionally be used to derive the key to decrypt the PAN data. The key identifier is an operational token or the key label of an operational token in key storage.

If you do not use the UKPT process or you specify the DUKPT-IP rule array keyword, the key token must contain the PIN-encrypting key to be used to encipher the output PIN block. The key algorithm must be DES, the key type must be OPINENC, and the key usage REFORMAT bit must be enabled.

If you use the UKPT process for the output PIN block by specifying the DUKPT-OP or DUKPT-BH rule array keyword, the key token must contain the key-generating key to derive the PIN-encrypting key. If you have also specified the OUTDUKPT keyword, the key will be used to derive the key to decrypt the PAN data. The key algorithm must be DES, the key type must be KEYGENKY, and the key usage UKPT bit 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.

PAN_key_identifier_length

Direction	Type
Input	Integer

Specifies the length of the *PAN_key_identifier* parameter in bytes. The value must be 64 if the PAN key management method keyword is STATIC; otherwise, the value is 0.

PAN_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the PAN data. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm must be DES, the key type must be CIPHER or DECIPHER, and the key must be a double-length key.

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

input_PIN_profile_length

Direction	Type
Input	Integer

Specifies the length of the *input_PIN_profile* parameter in bytes. The value is 24 if the profile does not contain a CKSN extension. The value is 48 when the CKSN extension is part of the profile.

input_PIN_profile

Direction	Type
Input	String

The 24 or 48 bytes input PIN profile. The profile consists of three 8-byte character strings with information defining the input PIN-block format and optionally, an additional 24 bytes containing the input CKSN extension.

PAN_data_length

Direction	Type
Input	Integer

When the input data mode keyword is CBC, this parameter specifies the length in bytes of the *PAN_data* parameter. The value must be 16.

When the input data mode keyword is VFPE, this parameter specifies the number of PAN digits. The value will be between 15 and 19 inclusive.

PAN_data

Direction	Type
Input	String

The enciphered primary account number (PAN) to be used to reformat the PIN block format. For VFPE mode, if the PAN contains an odd number of 4-bit digits, the data is left justified in the PAN variable and the rightmost 4 bits are ignored.

This service uses this data to recover the PIN from the PIN block if you specify the REFORMAT keyword and the input PIN profile specifies the ISO-0, VISA-4, or ISO-3 keyword for the PIN block format. If the output PIN profile specifies the ISO-0, VISA-4, or ISO-3 keyword for the PIN block format, the 12 rightmost digits of the PAN, excluding the check digit, are used to format the output PIN block.

input_PIN_block_length

Direction	Type
Input	Integer

Specifies the length of the *input_PIN_block* parameter in bytes. The value must be 8.

input_PIN_block

Direction	Type
Input	String

The 8-byte enciphered PIN block that contains the PIN to be processed.

output_PIN_profile_length

Direction	Type
Input	Integer

Specifies the length of the *output_PIN_profile* parameter in bytes. The value is 24 or 48.

output_PIN_profile

Direction	Type
Input	String

The 24 or 48 byte PIN profile for the output PIN block. The profile contains three 8-byte character strings with information defining the PIN-block format and optionally, an additional 24 bytes containing the output CKSN extension.

sequence_number

Direction	Type
Input	Integer

The 4 byte sequence number if the output PIN block format is 3621. Specify the integer value 99999. Otherwise, this parameter is ignored.

output_PIN_block_length

Direction	Type
Input/Output	Integer

Specifies the length of the *output_PIN_block* parameter in bytes. The value must be at least 8 bytes. On output, the value is updated with the actual number of bytes returned.

output_PIN_block

Direction	Type
Output	String

The 8 byte reformatted PIN block.

reserved1_length

Direction	Type
Input	Integer

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

reserved1

Direction	Type
Input	String

This field is ignored.

`reserved2_length`

Direction	Type
Input	Integer

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

`reserved2`

Direction	Type
Input	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.

Access control points

The **Encrypted PIN Translate Enhanced** access control in the domain role controls the function of this service. In addition, the **Encrypted PIN Translate – Reformat** access control must be enabled when the mode rule array keyword is REFORMAT.

If any of the rule array keywords that control UKPT key derivation (IN-DUKPT, OUTDUKPT, DUKPT-OP, DUKPT-IP, and DUKPT-BH) are specified, the **UKPT - PIN Verify**, **PIN Translate** access control must be enabled.

An enhanced PIN security mode is available for formatting an encrypted PIN block into IBM 3621 format or IBM 3624 format. To do this, you must enable the **PTR Enhanced PIN Security** access control point in the domain role. When activated, this mode limits checking of the PIN to decimal digits. No other PIN block consistency checking will occur.

An enhanced PIN security mode is available to implement restrictions required by the ANSI X9.8 PIN standard. To enforce these restrictions, you must enable the following control points in the domain role. See 'ANSI X9.8 PIN Restrictions' for a description of these controls.

- ANSI X9.8 PIN - Enforce PIN block restrictions
- ANSI X9.8 PIN - Allow only ANSI PIN blocks

Note: The **ANSI X9.8 PIN - Allow modification of PAN** access control is not applicable to this service as the PAN is not allowed to be modified.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service.

Encrypted PIN Translate Enhanced

Table 21. Encrypted PIN Translate Enhanced required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890		This service is not supported.
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	Requires the March 2016 or later licensed internal code (LIC).

Managing PKA Cryptographic Keys

PKA Key Import (CSNDPKI and CSNFPKI)

Use this service to import an external PKA private key token. (The private key must consist of a PKA private key and public key.) The secret values of the key may be:

- Clear
- Encrypted under a limited-authority DES importer key or an AES importer key if the *source_key_identifier* is an RSA token
- Encrypted under an AES Key Encryption Key if the *source_key_identifier* is an ECC token

This service can also import a clear PKA key. The PKA key token build service creates a clear PKA key token.

This service can also import an external trusted block token for use with the remote key export callable service.

Output of this service is an ICSF internal token of the RSA or ECC private key or trusted block.

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

Format

```
CALL CSNDPKI(
            return_code,
            reason_code,
            exit_data_length,
            exit_data,
            rule_array_count,
            rule_array,
            source_key_identifier_length,
            source_key_identifier,
            importer_key_identifier,
            target_key_identifier_length,
            target_key_identifier)
```

Parameters

`return_code`

Direction	Type
Output	Integer

The return code specifies the general result of the callable service.

`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.

`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. This may be 0 or 1.

`rule_array`

Direction	Type
Input	Character String

The *rule_array* parameter is an array of keywords. 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. The rule_array keywords are:

Table 22. Keywords for PKA Key Import

Keyword	Meaning
Token Type (optional)	
RSA	Specifies that the key token is for an RSA key. This is the default.

Table 22. Keywords for PKA Key Import (continued)

Keyword	**Meaning**
ECC	Specifies that the key token is for an ECC key.
Transport key type (optional)	
IKEK-AES	The *importer_key_identifier* is a AES key.
IKEK-DES	The *importer_key_identifier* is a DES key. This is the default.

source_key_identifier_length

Direction	**Type**
Input	Integer

The length of the source_key_identifier parameter. The maximum size is 3500 bytes.

source_key_identifier

Direction	**Type**
Input	String

Contains an external token or label of a PKA private key, without section identifier 0x14 (Trusted Block Information), or the trusted block in external form as produced by the Trusted Block Create (CSNDTBC and CSNETBC) service with the ACTIVATE keyword.

If a PKA private key without the section identifier 0x14 is passed in:

- There are no qualifiers. A retained key cannot be used.
- The key token must contain both public-key and private-key information. The private key can be in cleartext or it can be enciphered.
- This is the output of the PKA key generate (CSNDPKG) callable service or the PKA key token build (CSNDPKB) callable service.
- If encrypted, it was created on another platform.

If a PKA key token with section 0x14 is passed in:

- This service will be used to encipher the MAC key within the trusted block under the PKA master key instead of the IMP-PKA key-encrypting key.
- The importer_key_identifier must contain an IMP-PKA KEK in this case.

importer_key_identifier

Direction	**Type**
Input/Output	String

A variable-length field containing an AES or DES key identifier used to wrap the imported key. For RSA keys, this is either a DES limited authority transport key (IMP-PKA) or an AES transport key. For trusted blocks, this must be a DES limited authority transport key (IMP-PKA). For ECC keys, this must be an AES transport key.

This parameter contains one of the following:

- 64-byte label of a CKDS record that contains the transport key.
- 64-byte DES internal key token containing the transport key.
- a variable-length AES internal key token containing the transport key.

This parameter is ignored for clear tokens.

`target_key_identifier_length`

Direction	Type
Input/Output	Integer

The length of the *target_key_identifier* parameter. The maximum size is 3500 bytes. On output, and if the size is of sufficient length, the variable is updated with the actual length of the target_key_identifier field.

`target_key_identifier`

Direction	Type
Input/Output	String

This field contains the internal token or label of the imported PKA private key or a Trusted Block. If a label is specified on input, a PKDS record with this label must exist. The PKDS record with this label will be overwritten with imported key unless the existing record is a retained key. If the record is a retained key, the import will fail. A retained key record cannot be overwritten. If no label is specified on input, this field is ignored.

Restrictions

This service imports RSA keys of up to 4096 bits. However, the hardware configuration sets the limits on the modulus size of keys for digital signatures and key management; thus, the key may be successfully imported but fail when used if the limits are exceeded.

The *importer_key_identifier* is a limited-authority key-encrypting key.

CRT form tokens with a private section ID of X'05' cannot be imported into ICSF.

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.

An RSA modulus-exponent form token imported results in a X'06' format.

This service imports keys of any modulus size up to 4096 bits. However, the hardware configuration sets the limits on the modulus size of keys for digital signatures and key management; thus, the key may be successfully imported but fail when used if the limits are exceeded.

Access control points

The **PKA Key Import** access control point controls the function of this service. If the *source_key_token* parameter points to a trusted block, the **PKA Key Import - Import an External Trusted Block** access control point must also be enabled.

| To prevent the importing of key tokens which have the private key values in the
| clear, the **PKA Key Import – Disallow clear key import** must be enabled.

Required hardware

This table lists the required cryptographic hardware for each server type and describes restrictions for this callable service.

PKA Key Import

Table 23. PKA Key Import required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890	PCI X Cryptographic Coprocessor Crypto Express2 Coprocessor	RSA keys with moduli greater than 2048-bit length are not supported.
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).
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor Crypto Express3 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC).
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	ECC External token and Diffie-Hellman support requires the Sep. 2011 or later licensed internal code (LIC). Importing RSA keys wrapped with an AES transport key is not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	
IBM z13 IBM z13s	Crypto Express5 CCA Coprocessor	

PKA Key Token Build (CSNDPKB and CSNFPKB)

This callable service can be used create PKA key tokens. Specifically, it can be used to:

- Build external PKA key tokens containing unencrypted private key for ECC or RSA keys. You can use this token as input to the PKA Key Import service to obtain an operational internal token containing an enciphered private key.
- Build external RSA key tokens with the private key for use with the PKA Key Translate service.
- Build a skeleton token for ECC and RSA keys that you can use as input to the PKA Key Generate service.
- Build a public key token containing a clear unencrypted public key for an ECC or RSA keys and return the public key in a token format that other PKA services can use directly.

ECC key generation requires this information in the skeleton token:

- The key type: ECC.
- The type of curve: Prime or Brainpool.
- The size of P in bits: 192, 224, 256, 384 or 521 for Prime curves and 160, 192, 224, 256, 320, 384, or 521 for Brainpool curves.
- Key usage information.
- Optionally, application associated data.
- Optionally, a key-derivation section.

RSA key generation requires this following information in the skeleton token:

- In modulus-exponent form:
  - The length of the modulus n in bits (512-4096).
  - The length of the public exponent e (optional). There are restrictions on the value and length of the public exponent when the length of the modulus is greater than 2048.
  - The length of the private exponent d (optional).
  - The pubic exponent e (optional).
- In Chinese Remainder Theorem form:
  - The length of the modulus n in bits (512-4096).
  - The length of the public exponent e (optional).
  - The pubic exponent e (optional).
  - Other optional lengths.

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

Format

```
CALL CSNDPKB(
            return_code,
            reason_code,
            exit_data_length,
            exit_data,
            rule_array_count,
            rule_array,
            key_value_structure_length,
            key_value_structure,
            private_key_name_length,
            private_key_name,
            user_definable_associated_data_length,
            user_definable_associated_data,
|           key_derivation_data_length,
|           key_derivation_data,
            reserved_3_length,
            reserved_3,
            reserved_4_length,
            reserved_4,
            reserved_5_length,
            reserved_5,
            key_token_length,
            key_token)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service.

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.

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. Value must be 1, 2, 3, or 4.

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service. Table 24 lists the keywords. 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 PKA Key Token Build Control Information

Keyword	Meaning
Key Type (required)	
RSA-CRT	This keyword indicates building a token containing an RSA private key in the optimized Chinese Remainder Theorem (CRT) form. The parameter *key_value_structure* identifies the input key values, if supplied.
RSA-PRIV	This keyword indicates building a token containing both public and private RSA key information. The parameter *key_value_structure* identifies the input key values, if supplied.
RSA-PUBL	This keyword indicates building a token containing public RSA key information. The parameter *key_value_structure* identifies the input values, if supplied.
RSAMEVAR	This keyword is for creating a key token for an RSA public and private key pair in modulus-exponent form whose modulus is 512 bits or greater.
RSA-AESM	This keyword is for creating a key token for an RSA public and private key in modulus-exponent format. The object protection key is an AES key. The private key section id is X'30'

Table 24. Keywords for PKA Key Token Build Control Information (continued)

Keyword	Meaning
RSA-AESC	This keyword is for creating a key token for an RSA public and private key in Chinese-Remainder Theorem format. The object protection key is an AES key. The private key section id is X'31.
ECC-PAIR	This keyword indicates building a token containing both public and private ECC key information. The parameter *key_value_structure* identifies the input key values, if supplied.
ECC-PUBL	This keyword indicates building a token containing public ECC key information. The parameter *key_value_structure* identifies the input values, if supplied.
Key Usage Control (optional)	
KEY-MGMT	Indicates that a private key can be used in both the symmetric key import and the digital signature generate callable services.
KM-ONLY	Indicates that a private key can be used only in symmetric key distribution.
SIG-ONLY	Indicates that a private key cannot be used in symmetric key distribution. This is the default.
Translate Control (optional, only allowed with key types RSA-AESM, RSA-AESC, RSA-PRIV, RSAMEVAR, RSA-CRT, and ECC-PAIR and is valid with all key usage rules.)	
XLATE-OK	Specifies that the private key material can be translated.
NO-XLATE	Indicates key translation is not allowed. This is the default.
| ***ECC token version (one, optional. Only valid with token type ECC-PAIR.)***	
| ECC-VER0	Creates an ECC public and private key-pair containing a version 0 private key section. This is the default.
| ECC-VER1	Creates an ECC public and private key-pair containing a version 1 private key section.

`key_value_structure_length`

Direction	Type
Input	Integer

This is a segment of contiguous storage containing a variable number of input clear key values. The length depends on the key type parameter in the rule array and on the actual values input. The length is in bytes.

Table 25. Key Value Structure Length Maximum Values for Key Types

Key Type	Key Value Structure Maximum Value
RSA-CRT	3500
RSAMEVAR	3500
RSA-AESC	3500
RSA-AESM	3500
RSA-PRIV	648
RSA-PUBL	520
ECC-PAIR	207
ECC-PUBL	139

key_value_structure

Direction	Type
Input	String

This is a segment of contiguous storage containing a variable number of input clear key values and the lengths of these values in bits or bytes, as specified. The structure elements are ordered, of variable length, and the input key values must be right-justified within their respective structure elements and padded on the left with binary zeros. If the leading bits of the modulus are zero's, do not count them in the length. Table 26 defines the structure and contents as a function of key type.

Table 26. Key Value Structure Elements for PKA Key Token Build

Offset	Length (bytes)	Description
Key Value Structure: Optimized RSA, Chinese Remainder Theorem form (RSA-CRT, RSA-AESC)		
000	002	Modulus length in bits (512 to 4096). This is required.
002	002	Modulus field length in bytes, "nnn." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. This value must not exceed 512.
004	002	Public exponent field length in bytes, "eee." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service.
006	002	Reserved, binary zero.
008	002	Length of the prime number, p, in bytes, "ppp." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of p + q is 512 bytes.
010	002	Length of the prime number, q, in bytes, "qqq." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of p + q is 512 bytes.
012	002	Length of d_p, in bytes, "rrr." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of $d_p + d_q$ is 512 bytes.
014	002	Length of d_q, in bytes, "sss." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of $d_p + d_q$ is 512 bytes.

Table 26. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
016	002	Length of U, in bytes, "uuu." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of U is 512 bytes.
018	nnn	Modulus, n.
018 + nnn	eee	Public exponent, e. This is an integer such that 1<e<n. e must be odd. When you are building a *skeleton_key_token* to control the generation of an RSA key pair, the public key exponent can be one of these values: 3, 65537 (2^{16} + 1), or 0 to indicate that a full random exponent should be generated. The exponent field can be a null-length field if the exponent value is 0.
018 + nnn + eee	ppp	Prime number, p.
018 + nnn + eee + ppp	qqq	Prime number, q.
018 + nnn + eee + ppp + qqq	rrr	d_p = d mod(p-1).
018 + nnn + eee + ppp + qqq + rrr	sss	d_q = d mod(q-1).
018 + nnn + eee + ppp + qqq + rrr + sss	uuu	U = q^{-1}mod(p).
Key Value Structure: RSA Modulus-Exponent form (RSA-PRIV, RSA-PUBL, RSAMEVAR, RSA-AESM)		
000	002	Modulus length in bits. This is required. When building a skeleton token, the modulus length in bits must be greater than or equal to 512 bits.
002	002	Modulus field length in bytes, "XXX". TThis value must not exceed 512 when the RSA-PUBL, RSA-AESM, or RSAMEVAR keyword is used, and must not exceed 128 when the RSA-PRIV keyword is used. This service can build a key token for a public RSA key with a 4096-bit modulus length, or it can build a key token for a 1024-bit modulus length private key.

Table 26. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
004	002	Public exponent field length in bytes, “YYY”. This value must not exceed 512 when either the RSA-PUBL, RSA-AESM, or RSAMEVAR keyword is used, and must not exceed 128 when the RSA-PRIV keyword is used. This value can be zero if you are using the key token as a skeleton token in the PKA key generate verb. In this case, a random exponent is generated. To obtain a fixed, predetermined public key exponent, you can supply this field and the public exponent as input to the PKA key generate verb.
006	002	Private exponent field length in bytes, “ZZZ”. This field can be zero, indicating that private key information is not provided. This value must not exceed 128 bytes. This value can be zero if you are using the key token as a skeleton token in the PKA key generate verb.
008	XXX	Modulus, n. This is an integer such that $1 < n < 2^{**2048}$.The n is the product of p and q for primes p and q.
008 + XXX	YYY	RSA public exponent, e. This is an integer such that 1<e<n. e must be odd. When you are building a *skeleton_key_token* to control the generation of an RSA key pair, the public key exponent can be one of these values: 3, 65537 ($2^{16} + 1$), or 0 to indicate that a full random exponent should be generated. The exponent field can be a null-length field if the exponent value is 0.
008 + XXX + YYY	ZZZ	RSA secret exponent d. This is an integer such that 1<d<n. The value of d is e^{-1} mod(p-1)(q-1). e**-1 mod(p-1)(q-1); the product of e and d is 1 mod(p-1)(q-1). This can be a null-length field if you are using the key token as a skeleton token in the PKA key generate verb.
Key Value Structure: ECC Private/public key pair form (ECC-PAIR)		
000	001	Curve type **x'00'** Prime Curve **x'01'** Brainpool Curve
001	001	Reserved x'00'

Table 26. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
002	002	Length of p in bits **0x'00C0'** Prime P-192 **0x'00E0'** Prime P-224 **0x'0100'** Prime P-256 **0x'0180'** Prime P-384 **0x'0209'** Prime P-521 **0x'00A0'** Brain Pool P-160 **0x'00C0'** Brain Pool P-192 **0x'00E0'** Brain Pool P-224 **0x'0100'** Brain Pool P-256 **0x'0140'** Brain Pool P-320 **0x'0180'** Brain Pool P-384 **0x'0200'** Brain Pool P512.
004	002	ddd, This field is the length of the private key d value in bytes, This value can be zero if the key token is used as a skeleton key token in the PKA Key Generate callable service. The maximum value could be up to 66 bytes.
006	002	xxx, This field is the length of the public key Q value in bytes. This value can be zero if the key token is used as a skeleton key token in the PKA Key Generate callable service. The maximum value could be up to 133 bytes which includes one byte to indicate if the value is compressed.
008	ddd	Private key d
008 + ddd	xxx	Public Key value Q
Key value Structure: ECC Public form (ECC_PUBL)		
000	001	Curve type: **0x'00'** Prime Curve **0x'01'** Brain Pool Curve
001	001	Reserved x'00'

Table 26. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
002	002	Length of p in bits **0x'00C0'** Prime P-192 **0x'00E0'** Prime P-224 **0x'0100'** Prime P-256 **0x'0180'** Prime P-384 **0x'0209'** Prime P-521 **0x'00A0'** Brain Pool P-160 **0x'00C0'** Brain Pool P-192 **0x'00E0'** Brain Pool P-224 **0x'0100'** Brain Pool P-256 **0x'0140'** Brain Pool P-320 **0x'0180'** Brain Pool P-384 **0x'0200'** Brain Pool P512.
004	002	xxx, This field is the length of the public key Q value in bytes. This value can be zero if the key token is used as a skeleton key token in the PKA Key Generate callable service. The maximum value could be up to 133 bytes which includes a one byte value indicating compressed or uncompressed key value.
006	xxx	Public key value Q

Note:

1. All length fields are in binary.
2. All binary fields (exponent, lengths, modulus, and so on) are stored with the high-order byte field first. This integer number is right-justified within the key structure element field.
3. You must supply all values in the structure to create a token containing an RSA private key for input to the PKA key import service.

`private_key_name_length`

Direction	Type
Input	Integer

The length can be 0 or 64.

private_key_name

Direction	Type
Input	EBCDIC character

This field contains the name of a private key. The name must conform to ICSF label syntax rules. That is, allowed characters are alphanumeric, national (@,#,$) or period (.). The first character must be alphabetic or national. The name is folded to upper case and converted to ASCII characters. ASCII is the permanent form of the name because the name should be independent of the platform. The name is then cryptographically coupled with clear private key data prior to its encryption of the private key. Because of this coupling, the name can never change when the key token is already imported. The parameter is not valid with key type RSA-PUBL or ECC-PUBL.

user_definable_associated_data_length

Direction	Type
Input	Integer

The length of the *user_definable_associated_data* parameter.

Valid for Rule Array Key Type of ECC-PAIR with a maximum value of 100 and must be set to 0 for all other Rule Array Key Types.

user_definable_associated_data

Direction	Type
Input	String

The *user_definable_associated_data* parameter is a pointer to a string variable containing the associated data that will be placed following the IBM associated data in the token. The associated data is data whose integrity but not confidentiality is protected by a key wrap mechanism. It can be used to bind usage control information.

Valid for Rule Array Key Type of ECC-PAIR and is ignored for all others.

key_derivation_data_length

Direction	Type
Input	Integer

The length of the *key_derivation_data* parameter in bytes. When the token version keyword ECC-VER1 is specified, the value must be 0 or 4. Otherwise, the value must be 0.

key_derivation_data

Direction	Type
Input	String

The 4-byte key derivation data structure describing the key to be derived. This data will be used to create the optional key derivation section of an ECC key-token. Table 27 shows the contents of this structure.

Table 27. PKA Key Token Build key-derivation-data contents, ECC keys

Offset	Length (bytes)	Description
ECC key-derivation data, ECC-PAIR ECC-VER1		
000	001	Algorithm of the key to be derived: **X'01'** DES **X'02'** AES
001	001	Key type of the key to be derived: **X'01'** DATA **X'02'** EXPORTER **X'03'** IMPORTER **X'04'** CIPHER **X'05'** DECIPHER **X'06'** ENCIPHER **X'07'** CIPHERXI **X'08'** CIPHERXL **X'09'** CIPHERXO
002	002	Key bit length: 64, 128, 192, 256

reserved_3_length

Direction	Type
Input	Integer

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_3

Direction	Type
Input	String

The *reserved_3* parameter identifies a string that is reserved. The service ignores it.

reserved_4_length

Direction	Type
Input	Integer

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_4

Direction	Type
Input	String

The *reserved_4* parameter identifies a string that is reserved. The service ignores it.

reserved_5_length

Direction	Type
Input	Integer

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_5

Direction	Type
Input	String

The *reserved_5* parameter identifies a string that is reserved. The service ignores it.

key_token_length

Direction	Type
Input/Output	Integer

Length of the returned key token. The service checks the field to ensure it is at least equal to the size of the token to return. On return from this service, this field is updated with the exact length of the *key_token* created. On input, a size of 3500 bytes is sufficient to contain the largest *key_token* created.

key_token

Direction	Type
Output	String

The returned key token containing an unenciphered private or public key. The private key is in an external form that can be exchanged with different Common Cryptographic Architecture (CCA) PKA systems. You can use the public key token directly in appropriate ICSF signature verification or key management services.

Usage notes

If you are building a skeleton for use in a PKA Key Generate request to generate a retained PKA private key, you must build a private key name section in the skeleton token.

Required hardware

No cryptographic hardware is required by this callable service.

Reason codes for return code 8 (8)

Table 28 on page 64 lists reason codes returned from callable services that give return code 8.

Most of these reason codes indicate that the call to the service was unsuccessful. No cryptographic processing took place. Therefore, no output parameters were filled. Exceptions to this are noted in the descriptions.

Reason codes for return code 8 (8)

Table 28. Reason codes for return code 8 (8)

Reason Code Hex (Decimal)	Description
8BE (2238)	The supplied PIN profile has an invalid value. **User action**: Review the requirement of the service and correct the PIN profile.
8BF (2239)	The check digit compliance keyword denotes compliant check digit, but the input PAN does not have a compliant check digit.
8C3 (2243)	The CSNDEDH service was called and the token attributes in the skeleton token do not match those in the key-derivation section of the ECC private key token. **User action**: Provide a skeleton token with attributes that match the ECC private key token.
8C5 (2245)	The CSNDEDH service was called and the key-token pedigree / key source of the ECC private key did not meet requirements; for example, it was not randomly generated. **User action**: Supply an ECC private key token with the correct pedigree.
8C6 (2246)	The CSNDPKG service was passed an ECC private key token that is ill-formed. The token has an associated data section version of X'01' and is missing the IBM extended associated data required for a version X'01' token. **User action**: Supply an ECC private key token with the correct IBM extended associated data for a version X'01' token.
B82 (2946)	The maximum amount of plaintext/ciphertext that can be processed in the GCM mode by the CSNBSAD and CSNBSAE services was exceeded.
B83 (2947)	When deciphering ciphertext that had been created using Galois/Counter Mode (GCM) with the CSNBSAD service, the GCM tag provided did not match the data provided. No cleartext was returned. **User action**: Verify that the parameters provided (ciphertext, additional authenticated data, and tag) match those provided to, or returned from, the corresponding call to the CSNBSAE service.

Key token formats

An ECC key token is the concatenation of these sections:

- A token header:
  - An external header (first byte X'1E').
  - An internal header (first byte X'1F').
- An optional private key section (section identifier X'20').
- A public key section (section identifier X'21').
- An optional key-derivation section (section identifier X'23').

Table 29. ECC private-key section (X'20')

Offset (bytes)	Length (bytes)	Description
000	001	Section identifier: **X'20'** ECC private key.
001	001	Section version number: **X'00'** or **X'01'**
002	002	Section length in bytes: 92 + *kl* + *iead* + *uad* + *bb*.
004	001	Encrypted section wrapping method: **X'00'** Clear - section is unencrypted. **X'01'** AESKW. **X'02'** CBC Wrap - Other.

Table 29. ECC private-key section (X'20') (continued)

Offset (bytes)	Length (bytes)	Description
005	001	Hash used for wrapping: **X'01'** SHA224. **X'02'** SHA256.
006	002	Reserved, binary zero.
008	001	Key-usage and translation control flag: Management of symmetric keys and generation of digital signatures: **B'11xx xxxx'** Only key establishment (KM-ONLY). **B'10xx xxxx'** Both signature generation and key establishment (KEY-MGMT). **B'01xx xxxx'** Undefined. **B'00xx xxxx'** Only signature generation (SIG-ONLY). Translation control: **B'xxxx xx1x'** Private key translation is allowed (XLATE-OK). **B'xxxx xx0x'** Private key translation is not allowed (NO-XLATE). All other bits are reserved and must be zero.
009	001	Curve type: **X'00'** Prime curve. **X'01'** Brainpool curve.
010	001	Key format and security flag: External token: **X'40'** Unencrypted ECC private key subsection identifier. **X'42'** Encrypted ECC private key subsection identifier. Internal Token: **X'08'** Encrypted ECC private key subsection identifier. All other values are reserved and undefined.

Table 29. ECC private-key section (X'20') (continued)

Offset (bytes)	Length (bytes)	Description
011	001	Pedigree / Key source flag byte: **Version '00'** Reserved, binary zero. **Version '01'** External key-token: **X’00’** None / Clear **X’24’** Randomly generated Internal key-token: **X’00’** None / Clear **X’21’** Imported from cleartext **X’22’** Imported from ciphertext **X’24’** Randomly generated
012	002	Length of *p* in bits: **X'00C0'** Prime P-192. **X'00E0'** Prime P-224. **X'0100'** Prime P-256. **X'0180'** Prime P-384. **X'0209'** Prime P-521. **X'00A0'** Brainpool p-160. **X'00C0'** Brainpool P-192. **X'00E0'** Brainpool P-224. **X'0100'** Brainpool P-256. **X'0140'** Brainpool P-320. **X'0180'** Brainpool P-384. **X'0200'** Brainpool P-512.
014	002	Length of the IBM associated data.
016	008	Key verification pattern: External token: • For an encrypted private key, KEK verification pattern (KVP). • For a clear private key, binary zero. • For a skeleton, binary zero. Internal token: • For encrypted private key, master-key verification pattern (MKVP). • For a skeleton, binary zero.

Table 29. ECC private-key section (X'20') (continued)

Offset (bytes)	Length (bytes)	Description
024	048	Object Protection Key (OPK) data: External key-token: Reserved, binary zero. Internal key-token: The OPK consists of an 8-byte integrity check value (ICV) and length indicators, an 8-byte confounder, and a 256-bit AES key used with the AESKW algorithm to encrypt the ECC private key contained in an AESKW formatted section. The OPK is encrypted by the ECC master key using the AESKW algorithm.
072	002	Associated data length, *aa*.
074	002	Length of formatted section in bytes, *bb*.
		Start of the associated data section.
		Start of the IBM associated data section.
076	001	Associated data section version (**X'00'** or **X'01'**). Includes IBM associated data and user-definable associated data.
077	001	Length in bytes of the key label: *kl* (0 - 64).
078	002	Length in bytes of the IBM associated data, including key label and IBM extended associated data (≥ 16).
080	002	Length in bytes of the IBM extended associated data. Associated data section version: **'00'** 0 **'01'** 36
082	001	Length in bytes of the user-definable associated data: *uad* (0 - 155).
083	001	Curve type (see offset 009).
084	002	Length of *p* in bits (see offset 012).
086	001	Key-usage flag byte (see offset 008).
087	001	Key format and security flag byte (see offset 010).
088	004	Associated data section version: **'00'** Reserved, binary zero. **'01'** Pedigree / Key source flag byte (see offset 011).
092	*kl*	Optional key label.
092 + *kl*	*iead*	Optional IBM extended associated data.
		End of the IBM associated data section.
092+ *kl* + *iead*	*uad*	Optional user-definable associated data.
		End of associated data section.
092+ *kl* + *iead* + *uad*	*bbb*	Formatted section (payload), which includes private key d: • Clear-key section contains d. • Encrypted-key section contains d within the AESKW-wrapped payload.

IBM extended associated data section

The IBM extended associated data section consists of a series of TLV objects. The format of the TLV objects is shown in Table 30 on page 68.

Tag 'X'60' object contains the 32 byte SHA-256 hash of all of the optional sections concatenated to an ECC key-token. The object is described in Table 30.

Table 30. IBM extended associated data section TLV object

Offset	Length (bytes)	Description
000	001	Tag identifier: **X'60'** Optional section hash TLV object.
001	001	TLV object version number (X'00').
002	002	TLV object length in bytes (X'0024').
004	032	SHA-256 hash of all the optional sections that follow the public-key section, if any. Otherwise, binary 0.

Table 31. ECC key-derivation section (X'23')

Offset	Length (bytes)	Description
000	001	Section identifier: **X'23'** Key-derivation information.
001	001	Section version number: **X'00'**.
002	002	Section length in bytes: 8
004	001	Algorithm of the key to be derived: **X'01'** DES **X'02'** AES
005	001	Key type of the key to be derived: **X'01'** DATA **X'02'** EXPORTER **X'03'** IMPORTER **X'04'** CIPHER **X'05'** DECIPHER **X'06'** ENCIPHER **X'07'** CIPHERXI **X'08'** CIPHERXL **X'09'** CIPHERXO
006	002	Key bit length: 64, 128, 192, 256.

Access control points and callable services

The following tables list usage information using the following abbreviations:

AE Always enabled, cannot be disabled.

ED Enabled by default.

DD Disabled by default.

SC Usage of this access control point requires special consideration.

This table lists access control points that affect multiple services or require special consideration when enabling the access control point.

Table 32. Access control points affecting multiple services or requiring special consideration

Access control point name	Callable services	Notes	Usage
ANSI X9.8 PIN - Allow only ANSI PIN blocks	CSNBPTR / CSNEPTR andCSNBPTRE / CSNEPTRE,	See "ANSI X9.8 PIN Restrictions" for a description of this control.	DD, SC
ANSI X9.8 PIN - Enforce PIN block restrictions	CSNBCPA / CSNECPA, CSNBPTR / CSNEPTR, CSNBPTRE / CSNEPTRE, and CSNBSPN / CSNESPN	See "ANSI X9.8 PIN Restrictions" for a description of this control.	DD, SC
DUKPT - PIN Verify, PIN Translate	CSNBPVR / CSNEPVR, CSNBPTR / CSNEPTR and CSNBPTRE / CSNEPTRE,	When enabled, the listed services can use DUKPT key derivation.	ED
Enhanced PIN Security	CSNBCPE / CSNECPE, CSNBCPA / CSNECPA, CSNBEPG / CSNEEPG, CSNBPTR / CSNEPTR, CSNBPTRE / CSNEPTRE, CSNBPVR / CSNEPVR, and CSNBPCU / CSNEPCU	See "Enhanced PIN Security Mode" for a description of this control.	DD, SC

There are relationships between certain access control points. A controlling access control point is required to be enabled before subordinate access control points can enabled. The TKE workstation will enable the controlling access control point when a subordinate access control point is enabled.

- The Allow weak DES wrap of RSA access control point is only checked if the Prohibit weak wrap – Transport keys access control point is enabled.
- The ANSI X9.8 PIN - Allow modification of PAN and ANSI X9.8 PIN - Allow only ANSI PIN blocks access control points can only be enable when the ANSI X9.8 PIN - Enforce PIN block restrictions access control point is enabled.

This following table lists access control points that affect specific services indicated in the access control point name. There is a description of the usage of the access control point in the Usage Notes section of the callable service description.

Note: If the domain role has been changed via the TKE workstation, all new access control points are disabled by default.

Table 33. Access control points – Callable Services

Access control point name	Callable service	Usage
ECC Diffie-Hellman – Allow DERIV02	CSNDEDH / CSNFEDH	ED
Encrypted PIN Translate Enhanced	CSNBPTRE / CSNEPTRE	ED
Encrypted PIN Translate - Reformat	CSNBPTR / CSNEPTR and CSNBPTRE / CSNEPTRE	ED

Table 33. Access control points – Callable Services (continued)

Access control point name	Callable service	Usage
Key Test2 – AES, CMACZERO	CSNBKYT2 / CSNEKYT2	ED
Key Test2 – DES, CMACZERO	CSNBKYT2 / CSNEKYT2	ED
PKA Key Import – Disallow clear key import	CSNDPKI / CSNFPKI	DD, SC
Symmetric Algorithm Decipher – Galois/Counter mode AES	CSNBSAD / CSNESAD and CSNBSAD1 / CSNESAD1	ED
Symmetric Algorithm Encipher – Galois/Counter mode AES	CSNBSAE / CSNESAE and CSNBSAE1 / CSNESAE1	ED
Symmetric Algorithm Encipher - Secure AES	CSNBSAE / CSNESAE and CSNBSAE1 / CSNESAE1	ED
Symmetric Key Encipher/Decipher - Encrypted DES keys	CSNBSYD / CSNBSYE and CSNBSYD1 / CSNESYD1	ED
Symmetric Key Encipher/Decipher - Encrypted AES keys	CSNBSYD / CSNBSYE and CSNBSYD1 / CSNESYD1	ED

Chapter 5. Update of z/OS Cryptographic Services ICSF Administrator's Guide, SC14-7506-04, 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.

Controlling who can use cryptographic keys and services

Setting up profiles in the CSFSERV general resource class

Table 34. Resource names for ICSF callable services

Resource name	Callable service names	Callable service description
CSFPTRE	CSNBPTRE CSNEPTRE	Encrypted PIN Translate Enhanced

Callable services affected by key store policy

Table 35. Callable services and parameters affected by key store policy

ICSF callable service	31-bit name	Parameter checked
Encrypted PIN Translate Enhanced	CSNBPTRE	input_PIN_key_identifier output_PIN_key_identifier PAN_key_identifier

Printed in USA