z/OS

Cryptographic Services Integrated Cryptographic Service Facility DK AES PIN Support - APAR OA42246 Documentation for FMID HCR77A1

Contents

Chapter 1. Overview

This document update describes DK AES PIN support and contains alterations to information previously presented in the following books:

- *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SC14-7508-00
- *z/OS Cryptographic Services ICSF Administrator's Guide*, SC14-7506-00
- *z/OS Cryptographic Services ICSF System Programmer's Guide*, SC14-7507-00
- *z/OS Cryptographic Services ICSF Messages*, SC14-7509-00

The preceding books document capabilities provided by FMID HCR77A1 and support z/OS Version 2 Release 1.

Technical changes or additions related to DK AES PIN support in this document update are indicated by a vertical line to the left of the change.

These updates relate to the enhancements made to the ICSF product by the application of APAR OA42246.

Chapter 2. Update of z/OS Cryptographic Services ICSF Application Programmer's Guide, SC14-7508-00, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SC14-7508-00, for the DK AES PIN support provided by this APAR. Refer to this source document if background information is needed.

Introducing Symmetric Key Cryptography and Using Symmetric Key Callable Services

The German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) designed methods of creating, processing, and verifying PINs for its members. The methods use a PIN reference value (PRW) which is generated when a PIN is created or changed and used to verify the PIN supplied in a transaction. The methods are not dependent on a specific cryptographic algorithm, but DK has chosen the AES algorithm for its implementation.

AES Key Types

The following AES key types are added for the DK PIN methods. These key types can only be used with the DK PIN services. The symmetric key management services can be used to generate these key types. The Diversified Key Generate2 service can be used to derive these key types.

DKYGENKY
These keys are used to derive the other key types in this list.

MAC These keys are used to generate and verify message authentication codes. The CMAC algorithm is supported.

PINCALC
These keys are used to generate PINs.

PINPROT
These keys are used to encrypt and decrypt PIN blocks.

PINPRW
These keys are used to generate and verify PIN reference values.

Table 1. Descriptions of AES Key Types and service usage

AES Key Type	Usable with services
Fixed-length AES key-token, version X'04'	
DATA	Symmetric Algorithm Decipher, Symmetric Algorithm Encipher
Variable-length AES key-token, version X'05' *Cipher class (data operation keys)* These keys are used to cipher text.	
CIPHER	Symmetric Algorithm Decipher, Symmetric Algorithm Encipher, Ciphertext Translate2
Key-encrypting key class These keys are used to cipher other keys.	
EXPORTER	Key Generate2, Key Translate2, PKA Key Generate, Symmetric Key Export

Table 1. Descriptions of AES Key Types and service usage (continued)

AES Key Type	Usable with services
IMPORTER	Key Generate2, PKA Key Generate, Key Test2, Key Translate2, Restrict Key Attribute, Secure Key Import2, Symmetric Key Import2
MAC class These keys are used to generate and verify a message authentication code (MAC).	
MAC	DK Random PIN Generate, DK PIN Change, DK PAN Modify in Transaction
PIN class These keys are used in various financial-PIN processing services.	
PINCALC	
PINPROT	DK PIN Change, DK Random PIN Generate
PINPRW	DK PIN Change, DK PIN Verify, DK Random PIN Generate, DK PAN Modify in Transaction
Key generating class These keys are used to derive operational keys.	
DKYGENKY	Diversified Key Generate2

Variable-length symmetric key token

The variable-length symmetric key token uses a fixed-length payload. The payload is padded to a common length regardless of the length of the key being wrapped. There is a field in the token at offset 28 to indicate that the token has fixed-length payload. All new key tokens generated will have a fixed-length payload. HCR77A1 coexistence APAR OA42014 is required if the key tokens with the fixed-length payload are stored in the CKDS and that CKDS is shared with earlier releases of ICSF.

Common Cryptographic Architecture AES Key Management Services

The Diversified Key Generate2 callable service is a new service to derive AES keys.

Diversified Key Generate2 Callable Service (CSNBDKG2 and CSNEDKG2)

The Diversified Key Generate2 callable service generates a AES key based on the AES key-generating key, the processing method, and the parameter supplied. The key usage fields of the key-generating key also determines the type of target key that can be generated.

Financial Services for DK PIN Methods

This topic describes the new financial services that are based on the PIN methods of and meet the requirements specified by the German Banking Industry Committee, *Die Deutsche Kreditwirtschaft*, also known as DK. The intellectual property rights regarding the methods and specification belongs to the German Banking Industry Committee.

The following DK services are added:

- "DK PAN Modify in Transaction (CSNBDPMT and CSNEDPMT)" on page 5
- "DK PIN Change (CSNBDPC and CSNEDPC)" on page 5
- "DK PIN Verify (CSNBDPV and CSNEDPV)" on page 5

- "DK Random PIN Generate (CSNBDRPG and CSNEDRPG)"

DK PAN Modify in Transaction (CSNBDPMT and CSNEDPMT)

The DK PAN Modify in Transaction service is used to obtain a new PIN reference value (PRW) for an existing PIN when a merger has occurred and the account information has changed.

DK PIN Change (CSNBDPC and CSNEDPC)

The DK PIN Change service is used to allow a customer to change their PIN. The existing PIN and PIN reference value (PRW) and the new PIN are inputs and a new PRW is generated. Optionally, an encrypted PIN block can be generated or an encrypted script with the PIN embedded.

DK PIN Verify (CSNBDPV and CSNEDPV)

The DK PIN Verify service is used to verify the PIN in a transaction. The account, the card data, and PRW are used to verify the PIN.

DK Random PIN Generate (CSNBDRPG and CSNEDRPG)

The DK Random PIN Generate service generates a random PIN and calculates the PRW. The account and card data are used to generate the PRW. An optional encrypted PIN block is generated for printing.

Managing Symmetric Cryptographic Keys

This topic provides the new and updated callable services that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods:

- "Diversified Key Generate2 Callable Service (CSNBDKG2 and CSNEDKG2)"
- "Key Generate2 (CSNBKGN2 and CSNEKGN2)" on page 10
- "Key Part Import2 (CSNBKPI2 and CSNEKPI2)" on page 22
- "Key Test2 (CSNBKYT2 and CSNEKYT2)" on page 23
- "Key Token Build2 (CSNBKTB2 and CSNEKTB2)" on page 24
- "Key Translate2 (CSNBKTR2 and CSNEKTR2)" on page 52
- "Restrict Key Attribute (CSNBRKA and CSNERKA)" on page 59
- "Secure Key Import2 (CSNBSKI2 and CSNESKI2)" on page 60
- "Symmetric Key Export (CSNDSYX and CSNFSYX)" on page 65
- "Symmetric Key Import2 (CSNDSYI2 and CSNFSYI2)" on page 66

Diversified Key Generate2 Callable Service (CSNBDKG2 and CSNEDKG2)

The diversified key generate2 service generates an AES key based on a function of a key-generating key, the process rule, and data that you supply.

To use this service, specify:

- The rule array keyword to select the diversification process.
- The operational AES key-generating key from which the diversified keys are generated.
  - Key usage field 1 determines the type of key that is generated and restricts the use of this key to the key-diversification process.
  - Key usage field 2 contains a flag to determine how key usage fields 3 through 6 control the key usage fields of the generated key.

- When the flag is on, the key usage fields of the DKYGENKY must be equal (KUF-MBE or 'KUF – must be equal') to the key usage fields of the generated key.
- When the flag is off, the key usage fields of the DKYGENKY limit the values of the key usage fields of the generated key (KUF-MBP, or 'KUF must be permitted').

For the service to be valid, the generated key cannot have usage that is not enabled in the DKYGENKY key. The UDX-ONLY bit is always treated as 'must be equal'.

– Key usage fields 3 through 6 in the key generating key indicate the key usage attributes for the key to be generated.

Note: The only exception to this rule is when the type of key to diversify is D-ALL.

- The data and length of data used in the diversification process.
- The AES key token with a suitable key usage field for receiving the diversified key.

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

Format

```
CALL CSNBDKG2(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             generating_key_identifier_length,
             generating_key_identifier,
             derivation_data_length,
             derivation_data,
             reserved1_length,
             reserved1,
             reserved2_length,
             reserved2,
             generated_key_identifier1_length,
             generated_key_identifier1,
             generated_key_identifier2_length,
             generated_key_identifier2)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. “ICSF and TSS Return and Reason Codes” on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned

to it that indicate specific processing problems. "ICSF and TSS Return and Reason Codes" on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFF' (2 gigabytes). The data is identified in the *exit_data* parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

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

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 2. Rule array keywords for diversified key generate2

Keyword	Meaning
Diversification Process (required)	
SESS-ENC	A session key is created by enciphering a 16-byte diversification value with the *k*-bit AES key-generating key to produce a *k*-bit AES session key using the AES algorithm in ECB mode, where *k* is 128, 192 or 256 bits.

generating_key_identifier_length

Direction	Type
Input	Integer

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

generating_key_identifier

Direction	Type
Input/Output	String

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

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

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

derivation_data_length

Direction	Type
Input	Integer

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

derivation_data

Direction	Type
Input	String

The derivation data to be used in the key generation process. This data is often referred to as the diversification data. For SESS-ENC, the derivation data is 16-bytes long. Note that if SESS-ENC is specified and the length of the key generating key is 192 bits or 256 bits, the data is manipulated in conformance with the EMV Common Session Key Derivation Option.

reserved1_length

Direction	Type
Input	Integer

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

reserved1

Direction	Type
Input	String

This field is ignored.

reserved2_length

Direction	Type
Input	Integer

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

reserved2

Direction	Type
Input	String

This field is ignored.

generated_key_identifier1_length

Direction	Type
Input/Output	Integer

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

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

generated_key_identifier1

Direction	Type
Input/Output	String

The buffer for the generated key token.

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

The key usage fields in the generated key must meet the requirements (KUF 'must be equal' or 'must be permitted') of the corresponding key usage fields in the generating key unless D-ALL is specified in the generating key. D-ALL permits the derivation of several different keys. A flag bit in the DKYGENKY key-usage field 2 determines whether the key-usage field level of control is KUF-MBE or KUF-MBP.

On output, the buffer contains the generated key token.

generated_key_identifier2_length

Direction	Type
Input/Output	Integer

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

generated_key_identifier2

Direction	Type
Input/Output	String

This field is ignored.

Usage Notes

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

Access Control Points

The **Diversified Key Generate2 – AES EMV1 SESS** access control point in the domain role controls the function of this service.

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

Required Hardware

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

Table 3. Diversified key generate2 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	Crypto Express3 Coprocessor	Requires the November 2013 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	Requires the September 2013 or later licensed internal code (LIC).

Key Generate2 (CSNBKGN2 and CSNEKGN2)

Use the Key Generate2 callable service to generate either one or two keys of any type. This callable service does not produce keys in clear form and all keys are returned in encrypted form. When two keys are generated, each key has the same clear value, although this clear value is not exposed outside the secure cryptographic feature.

This service returns variable-length CCA key tokens and uses the AESKW wrapping method.

This service supports HMAC and AES keys. Operational keys will be encrypted under the AES master key.

Some key types are not directly supported by this service because there is no default key usage value. These key types can be generated by using the TOKEN keyword and a skeleton token from the Key Token Build2 service. These AES key types require TOKEN be used: DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW.

The callable service name for AMODE(64) is CSNEKGN2.

Format

```
CALL CSNBKGN2(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             clear_key_bit_length,
```

```
key_type_1,
key_type_2,
key_name_1_length,
key_name_1,
key_name_2_length,
key_name_2,
user_associated_data_1_length,
user_associated_data_1,
user_associated_data_2_length,
user_associated_data_2,
key_encrypting_key_identifier_1_length,
key_encrypting_key_identifier_1,
key_encrypting_key_identifier_2_length,
key_encrypting_key_identifier_2,
generated_key_identifier_1_length,
generated_key_identifier_1,
generated_key_identifier_2_length,
generated_key_identifier_2 )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. "ICSF and TSS Return and Reason Codes" on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. "ICSF and TSS Return and Reason Codes" on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFF' (2 gigabytes). 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 2, 3 or 4.

rule_array

Direction	Type
Input	String

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

Table 4. Keywords for Key Generate2 Control Information

Keyword	Meaning
Token algorithm (required)	
HMAC	Specifies to generate an HMAC key token.
AES	Specifies to generate an AES key token.
Key Form (required) The first two characters refer to **key_type_1**. The next two characters refer to **key_type_2**. See the Usage Notes section for further details.	
EX	One key that can be sent to another system.
EXEX	A key pair; both keys to be sent elsewhere, possibly for exporting to two different systems. Both keys have the same clear value.
IM	One key that can be locally imported. The key can be imported onto this system to make it operational at another time.
IMEX	A key pair to be imported; one key to be imported locally and one key to be sent elsewhere. Both keys have the same clear value.
IMIM	A key pair to be imported; both keys to be imported locally at another time. Both keys have the same clear value.
OP	One operational key. The key is returned to the caller in operational form to be used locally.
OPEX	A key pair; one key that is operational and one key to be sent elsewhere. Both keys have the same clear value.
OPIM	A key pair; one key that is operational and one key to be imported locally at another time. Both keys have the same clear value.
OPOP	A key pair; either with the same key type with different associated data or complementary key types. Both keys have the same clear value.
Payload format version for generated_key_identifier_1 (one, optional) **Note:** This keyword overrides payload format version of any corresponding skeleton token.	
V0PYLDK1	Build a token with the old variable-length payload format for the generated_key_identifier_1 parameter. This is the default for AES CIPHER, EXPORTER, and IMPORTER key types and is only valid with those key types.

Table 4. Keywords for Key Generate2 Control Information (continued)

Keyword	Meaning
V1PYLDK1	Build a token with the new fixed-length payload format for the generated_key_identifier_1 parameter. This is the default for AES MAC, PINPROT, PINCALC, PINPRW, and DKYGENKY key types. Not valid with the HMAC MAC key type.
Payload format version for generated_key_identifier_2 (one, optional) **Note:** This keyword overrides payload format version of any corresponding skeleton token.	
V0PYLDK2	Build a token with the old variable-length payload format for the generated_key_identifier_2 parameter. This is the default for AES CIPHER, EXPORTER, and IMPORTER key types and is only valid with those key types.
V1PYLDK2	Build a token with the new fixed-length payload format for the generated_key_identifier_2 parameter. This is the default for AES MAC, PINPROT, PINCALC, PINPRW, and DKYGENKY key types. Not valid with the HMAC MAC key type.

clear_key_bit_length

Direction	Type
Input	Integer

The size (in bits) of the key to be generated.

- For the HMAC algorithm, this is a value between 80 and 2048, inclusive.
- For the AES algorithm, this is a value of 128, 192, or 256.

When *key_type_1* or *key_type_2* is TOKEN, this value overrides the key length contained in *generated_key_identifier_1* or *generated_key_identifier_2*, respectively.

key_type_1

Direction	Type
Input	String

Use the *key_type_1* parameter for the first, or only, key that you want generated. The keyword must be left-justified and padded with blanks. Valid type combinations depend on the key form, and are documented in Table 7 on page 18 and Table 8 on page 18.

The 8-byte keyword for the *key_type_1* parameter can be one of the following:

Table 5. Keywords and associated algorithms for key_type_1 parameter

Keyword	Algorithm
CIPHER	AES
EXPORTER	AES
IMPORTER	AES
MAC	HMAC
MACVER	HMAC
Specify the keyword TOKEN when supplying a key token in the *generated_key_identifier_1* parameter.	

If *key_type_1* is TOKEN, the associated data in the *generated_key_identifier_1* parameter is examined to derive the key type.

key_type_2

Direction	Type
Input	String

Use the *key_type_2* parameter for a key pair, which is shown in Table 8 on page 18. The keyword must be left-justified and padded with blanks. Valid type combinations depend on the key form.

The 8-byte keyword for the *key_type_2* parameter can be one of the following:

Table 6. Keywords and associated algorithms for key_type_2 parameter

Keyword	Algorithm
CIPHER	AES
EXPORTER	AES
IMPORTER	AES
MAC	HMAC
MACVER	HMAC
Specify the keyword TOKEN when supplying a key token in the *generated_key_identifier_2* parameter.	

If *key_type_2* is TOKEN, the associated data in the *generated_key_identifier_2* parameter is examined to derive the key type.

When only one key is being generated, this parameter is ignored.

key_name_1_length

Direction	Type
Input	Integer

The length of the *key_name* parameter for *generated_key_identifier_1*. Valid values are 0 and 64 bytes.

key_name_1

Direction	Type
Input	String

A 64-byte key store label to be stored in the associated data structure of *generated_key_identifier_1*.

key_name_2_length

Direction	Type
Input	Integer

The length of the *key_name* parameter for *generated_key_identifier_2*. Valid values are 0 and 64 bytes.

When only one key is being generated, this parameter is ignored.

key_name_2

Direction	Type
Input	String

A 64-byte key store label to be stored in the associated data structure of *generated_key_identifier_2*.

When only one key is being generated, this parameter is ignored.

user_associated_data_1_length

Direction	Type
Input	Integer

The length of the user-associated data parameter for *generated_key_identifier_1*. The valid values are 0 to 255 bytes.

user_associated_data_1

Direction	Type
Input	String

User-associated data to be stored in the associated data structure for *generated_key_identifier_1*.

user_associated_data_2_length

Direction	Type
Input	Integer

The length of the user-associated data parameter for *generated_key_identifier_2*. The valid values are 0 to 255 bytes.

When only one key is being generated, this parameter is ignored.

user_associated_data_2

Direction	Type
Input	String

User associated data to be stored in the associated data structure for *generated_key_identifier_2*.

When only one key is being generated, this parameter is ignored.

key_encrypting_key_identifier_1_length

Direction	Type
Input	Integer

The length of the buffer for *key_encrypting_key_identifier_1* in bytes. When the Key Form rule is OP, OPOP, OPIM, or OPEX, this length must be 0. When the Key Form rule is EX, EXEX, IM, IMEX, or IMIM, the value must be between the actual length of the token and 725 bytes when *key_encrypting_key_identifier_1* is a token.

The value must be 64 bytes when *key_encrypting_key_identifier_1* is a label.

key_encrypting_key_identifier_1

Direction	Type
Input/Output	String

When *key_encrypting_key_identifier_1_length* is 0, this parameter is ignored. Otherwise, *key_encrypting_key_identifier_1* contains an internal key token containing the AES importer or exporter key-encrypting key, or a key label.

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

key_encrypting_key_identifier_2_length

Direction	Type
Input	Integer

The length of the buffer for *key_encrypting_key_identifier_2* in bytes. When the Key Form rule is OPOP, this length must be 0. When the Key Form rule is EXEX, IMEX, IMIM, OPIM, or OPEX, the value must be between the actual length of the token and 725 when *key_encrypting_key_identifier_2* is a token. The value must be 64 when *key_encrypting_key_identifier_2* is a label.

When only one key is being generated, this parameter is ignored.

key_encrypting_key_identifier_2

Direction	Type
Input/Output	String

When *key_encrypting_key_identifier_2_length* is 0, this parameter is ignored. Otherwise, *key_encrypting_key_identifier_2* contains an internal key token containing the AES importer or exporter key-encrypting key, or a key label.

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

When only one key is being generated, this parameter is ignored.

generated_key_identifier_1_length

Direction	Type
Input/Output	Integer

On input, the length of the buffer for the *generated_key_identifier_1* parameter in bytes. The maximum value is 900 bytes.

On output, the parameter will hold the actual length of the *generated_key_identifier_1*.

generated_key_identifier_1

Direction	Type
Input/Output	String

The buffer for the first generated key token.

On input, if you specify a key_type_1 of TOKEN, then the buffer contains a valid key token of the key type you want to generate. The key token must be left justified in the buffer. See *key_type_1* for a list of valid key types.

On output, the buffer contains the generated key token.

generated_key_identifier_2_length

Direction	Type
Input/Output	Integer

On input, the length of the buffer for the *generated_key_identifier_2* in bytes. The maximum value is 900 bytes.

On output, the parameter will hold the actual length of the *generated_key_identifier_2*.

When only one key is being generated, this parameter is ignored.

generated_key_identifier_2

Direction	Type
Input/Output	String

The buffer for the second generated key token.

On input, if you specify a *key_type_2* of TOKEN, then the buffer contains a valid key token of the key type you want to generate. The key token must be left justified in the buffer. See *key_type_2* for a list of valid key types.

On output, the buffer contains the generated key token.

When only one key is being generated, this parameter is ignored.

Usage Notes

The key forms are defined as follows:

Operational (OP)
The key value is enciphered under a master key. The result is placed into an internal key token. The key is then operational at the local system.

Importable (IM)
The key value is enciphered under an importer key-encrypting key. The result is placed into an external key token. The corresponding *key_encrypting_key_identifier_x* parameter must contain an AES IMPORTER key token or label.

Exportable (EX)
The key value is enciphered under an exporter key-encrypting key. The result is placed into an external key token. The corresponding *key_encrypting_key_identifier_x* parameter must contain an AES EXPORTER key token or label.

The following tables list the valid key type and key form combinations and required access control points.

The key usage attributes that are shown are those that are required. Key usage attributes that are not shown are optional. When key usage attributes are not the default values for a key type, a skeleton key token with the desired attributes must be supplied and a key type must be **TOKEN**.

Key Generate2

Table 7 lists all key types that can be generated as a single key. The **Key Generate2 - OP** access control point must be enabled.

The key types marked with an asterisk (*) must be requested through the specification of a proper key usage field in a key token and the use of the **TOKEN** keyword.

Table 7. Key Generate2 valid key type and key form for one AES or HMAC key

key_type_1 (key usage)	Key Form OP, IM, EX	Notes
CIPHER (ENCRYPT, DECRYPT)	X	If you supply a skeleton token, the key usage must allow decryption and encryption.
*DKYGENKY (D-ALL)	X	
*DKYGENKY (D-CIPHER)	X	If you supply a skeleton token, the key usage for the derived key must allow decryption and encryption.
*DKYGENKY (D-MAC)	X	If you supply a skeleton token, the key usage for the derived key may not specify ONLY generate or ONLY verify.
*DKYGENKY (D-PCALC)	X	
HMAC MAC (GENERATE)	X	If you supply a skeleton token, the key usage may not specify ONLY generate or ONLY verify.
AES MAC (GENERATE)	X	The key usage in the skeleton may not specify ONLY generate or ONLY verify.

Table 8 lists all pairs of keys that can be generated and the key forms that are allowed. The key forms that are marked with an 'X' required the **Key Generate2 - Key set** access control point to be enabled. The key forms marked with an 'E' required the **Key Generate2 - Key set extended** access control point to be enabled.

The key types marked with an asterisk (*) must be requested through the specification of a proper key usage field in a key token and the use of the **TOKEN** keyword.

Table 8. Key Generate2 Valid key type and key forms for two AES or HMAC keys

key_type_1 (key usage)	key_type_2 (key usage)	Key Form OPOP OPIM IMIM	Key Form OPEX	Key Form EXEX	Key Form IMEX
CIPHER (DECRYPT, ENCRYPT)	CIPHER (DECRYPT, C-XLATE)	X	X	X	X
CIPHER (DECRYPT, ENCRYPT)	CIPHER (DECRYPT, ENCRYPT, C-XLATE)	X	X	X	X
CIPHER (DECRYPT, ENCRYPT)	CIPHER (ENCRYPT, C-XLATE)	X	X	X	X
CIPHER (DECRYPT)	CIPHER (ENCRYPT, C-XLATE)	X	X	X	X
CIPHER (DECRYPT, C-XLATE)	CIPHER (DECRYPT, ENCRYPT)	X	X	X	X
CIPHER (DECRYPT, ENCRYPT, C-XLATE)	CIPHER (DECRYPT, ENCRYPT)	X	E	X	E
CIPHER (ENCRYPT, C-XLATE)	CIPHER (DECRYPT, ENCRYPT)	X	E	X	E

Table 8. Key Generate2 Valid key type and key forms for two AES or HMAC keys (continued)

key_type_1 (key usage)	key_type_2 (key usage)	Key Form OPOP OPIM IMIM	Key Form OPEX	Key Form EXEX	Key Form IMEX
CIPHER (ENCRYPT, C-XLATE)	CIPHER (DECRYPT)	X	E	X	E
CIPHER (DECRYPT, C-XLATE)	CIPHER (DECRYPT)	X	E	X	E
CIPHER (DECRYPT, ENCRYPT, C-XLATE)	CIPHER (DECRYPT, ENCRYPT, C-XLATE)		E	X	E
CIPHER (DECRYPT, C-XLATE)	CIPHER (ENCRYPT, C-XLATE)		E	X	E
CIPHER (ENCRYPT, C-XLATE)	CIPHER (DECRYPT, C-XLATE)		E	X	E
*DKYGENKY	*DKYGENKY	X	X	X	X
EXPORTER	IMPORTER		X	X	X
IMPORTER	EXPORTER		X	X	X
MAC (GENERATE)	MAC (GENERATE)	X	X	X	X
MAC (GENERATE)	MAC (VERIFY)	X	X	X	X
MAC (GENERATE)	MAC (GENONLY)	X	X	X	X
MAC (GENONLY)	MAC (GENERATE)	X	X	X	X
MAC (GENONLY)	MAC (VERIFY)	X	X	X	X
MAC (VERIFY)	MAC (GENERATE)	X	X	X	X
MAC (VERIFY)	MAC (GENONLY)	X	X	X	X

See Table 12 on page 21 for an explanation of the differences between **E** as compared to **X**.

Note: A pair of DKYGENKY keys can be used to diversify a pair of keys with different key types and key usage attributes. The combination of key types and key usage attributes that can be diversified must meet the requirements of using the KGN2 verb to generate those same keys. A DKYGENKY key with D-ALL usage can only be paired with a DKYGENKY key with D-ALL usage.

For keys for the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN method, a key token with the proper key-usage values must be supplied. The key type 1 and 2 are TOKEN. Table 9 shows the valid key pairs. Access control points are required to be enabled for the generation of these keys.

Table 9. Valid key pairs that can be generated and their required access points

Access Control Point	Table identifier
Key Generate2 - OP	O
Key Generate2 - Key set	X
Key Generate2 - DK PIN key set	D
Key Generate2 - DK PIN admin1 key PINPROT	D1P
Key Generate2 - DK PIN admin1 key MAC	D1M
Key Generate2 - DK PIN print key	DP

Table 9. Valid key pairs that can be generated and their required access points (continued)

Access Control Point	Table identifier
Key Generate2 - DK PIN admin2 key MAC	D2

Table 10. Key type and key form keywords for AES keys - DK PIN methods

key type 1 (key usage)	key type 2 (key usage)	Key Form OPOP OPIM IMIM	Key Form OPEX IMEX	Key Form EXEX	Key Form OP EX IM
MAC(GENONLY, DKPINOP)	MAC(VERIFY, DKPINOP)	D	X		
MAC(VERIFY, DKPINOP)	MAC(GENONLY, DKPINOP)	D			
MAC(GENONLY, DKPINAD1)	MAC(VERIFY, DKPINAD1)	D1M	D1M		
MAC(VERIFY, DKPINAD1)	MAC(GENONLY, DKPINAD1)	D1M			
MAC(GENONLY, DKPINAD2)	MAC(VERIFY, DKPINAD2)	D	D2M		
MAC(VERIFY, DKPINAD2)	MAC(GENONLY, DKPINAD2)	D			
PINCALC(GENONLY, DKPINOP)					O
PINPROT(ENCRYPT, DKPINOP)	PINPROT(DECRYPT, DKPINOP)	D	X		
PINPROT(DECRYPT, DKPINOP)	PINPROT(ENCRYPT, DKPINOP)	D			
PINPROT(ENCRYPT, DKPINAD1)	PINPROT(DECRYPT, DKPINAD1)	D	D1P		
PINPROT(DECRYPT, DKPINAD1)	PINPROT(ENCRYPT, DKPINAD1)	D			
PINPROT(ENCRYPT, DKPINOPP)	CIPHER(DECRYPT)	D	DP		
CIPHER(DECRYPT)	PINPROT(ENCRYPT, DKPINOPP)	D			
PINPRW(GENONLY, DKPINOP)	PINPRW(VERIFY, DKPINOP)	X	X		
PINPRW(VERIFY, DKPINOP)	PINPRW(GENONLY, DKPINOP)	X			

The strength of the key-encrypting key used to wrap a generated key will affect the results of the service. The resulting return code and reason code when using a key-encrypting key that is weaker than the key being generated depends on the **Prohibit weak wrapping - Transport keys** and **Warn when weak wrap - Transport keys** access control points:

- If the **Prohibit weak wrapping - Transport keys** access control point is disabled, the key strength requirement will not be enforced. Using a weaker key will result in return code 0 with a non-zero reason code if the **Warn when weak wrap - Transport keys** access control point is enabled. Otherwise, a reason code of 0 will be returned.
- If the **Prohibit weak wrapping - Transport keys** access control point is enabled, the key strength requirement will be enforced, and attempting to use a weaker key will result in return code 8.

For AES keys, the AES KEK must be at least as strong as the key being generated to be considered sufficient strength.

For HMAC keys, the AES KEK must be sufficient strength as described in the following table.

Table 11. AES KEK strength required for generating an HMAC key under an AES KEK

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

Access Control Points

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

Table 12. Required access control points for Key Generate2

Access Control Point	Function control
Key Generate2 - OP	Key Form OP, EX, IM.
Key Generate2 - Key set	The key-form and key-type combinations shown with an X in Table 8 on page 18 and Table 10 on page 20.
Key Generate2 - Key set extended	The key-form and key-type combinations shown with an E in Table 10 on page 20.
Key Generate2 - DK PIN key set	The key-form and key-type combinations shown with an D in Table 10 on page 20.
Key Generate2 - DK PIN Admin1 Set PINPROT	The key-form and key-type combinations shown with an D1P in Table 10 on page 20.
Key Generate2 - DK PIN Admin1 Set MAC	The key-form and key-type combinations shown with an D1M in Table 10 on page 20.
Key Generate2 - DK PIN Print Set	The key-form and key-type combinations shown with an DP in Table 10 on page 20.
Key Generate2 - DK PIN Admin2 Set MAC	The key-form and key-type combinations shown with an D2 in Table 10 on page 20.
Prohibit weak wrapping - Transport keys	Prohibit wrapping a key with a weaker key.
Warn when weak wrap - Transport keys	Issue a non-zero reason code when using a weak wrapping key.

Required Hardware

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

Table 13. Key Generate2 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.

Table 13. Key Generate2 required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	DK AES PIN support requires the November 2013 or later licensed internal code (LIC). V0PYLDK1, V1PYLDK1, V0PYLDK2, and V1PYLDK2 keywords require the November 2013 or later licensed internal code (LIC). 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).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN support requires the September 2013 or later licensed internal code (LIC). V0PYLDK1, V1PYLDK1, V0PYLDK2, and V1PYLDK2 keywords require the September 2013 or later licensed internal code (LIC).

Key Part Import2 (CSNBKPI2 and CSNEKPI2)

Use the Key Part Import2 callable service to combine, by exclusive ORing, the clear key parts of any key type and return the combined key value either in a variable-length internal token or as an update to the CKDS.

Prior to using the key part import2 service for the first key part, you must use the Key Token Build2 service to create the internal key token into which the key will be imported. Subsequent key parts are combined with the first part in internal token form or as a label from the CKDS.

On each call to Key Part Import2 (except with the COMPLETE keyword), specify the number of bits to use for the clear key part. Place the clear key part in the *key_part* parameter, and specify the number of bits using the *key_part_length* variable. Any extraneous bits of *key_part* data will be ignored.

Consider using the Key Test2 callable service to ensure a correct key value has been accumulated prior to using the COMPLETE option to mark the key as fully operational.

The callable service name for AMODE(64) is CSNEKPI2.

Required Hardware

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

Table 14. Key Part Import2 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 Part Import2 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). 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).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN support requires the September 2013 or later licensed internal code (LIC).

Key Test2 (CSNBKYT2 and CSNEKYT2)

Use this callable service to generate or verify a secure, cryptographic verification pattern for 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.

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

Required Hardware

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

Table 15. 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.
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 keyword is not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 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).

Key Token Build2 (CSNBKTB2 and CSNEKTB2)

Use the Key Token Build2 callable service to build a variable-length CCA symmetric key token in application storage from information that you supply. A clear key token built by this service can be used as input for the Key Test2 callable service. A skeleton token built by this service can be used as input for the Diversified Key Generate2, Key Generate2, Key Part Import2, and Secure Key Import2 callable services.

This service will build internal or external HMAC and AES tokens, both as clear key tokens and as skeleton tokens containing no key.

The callable service name for AMODE(64) is CSNEKTB2.

Format

```
CALL CSNBKTB2(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             clear_key_bit_length,
             clear_key_value,
             key_name_length,
             key_name,
             user_associated_data_length,
             user_associated_data,
             token_data_length,
             token_data,
             service_data_length,
             service_data,
             target_key_token_length,
             target_key_token )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. “ICSF and TSS Return and Reason Codes” on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. “ICSF and TSS Return and Reason Codes” on page 120 lists the reason codes.

exit_data_length

Direction	Type
Ignored	Integer

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

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you supplied in the *rule_array* parameter. The minimum value is 3, and the maximum value is 34.

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 16. Keywords for Key Token Build2 Control Information

Keyword	Meaning
Token type (one required)	
EXTERNAL	Specifies to build an external key token.
INTERNAL	Specifies to build an internal key token.
Token algorithm (one required)	
AES	Specifies to build an AES key token.
HMAC	Specifies to build an HMAC key token.
Key status (one, optional)	
KEY-CLR	Specifies to build the key token with a clear key value. This creates a key token that can be used with the Key Test2 service to generate a verification pattern for the key value.
NO-KEY	Specifies to build the key token without a key value. This creates a skeleton key token that can later be supplied to the Key Generate2 service. This is the default.
Payload version (one, optional)	
V0PYLD	Build a token with the old variable-length payload format for the target token. This is the default for AES CIPHER, EXPORTER, IMPORTER key types and is only valid with those key types.

Table 16. Keywords for Key Token Build2 Control Information (continued)

Keyword	Meaning
V1PYLD	Build a token with the new fixed-length payload format for the target token. This is the default for AES MAC, PINPROT, PINCALC, PINPRW, and DKYGENKY key types. Not valid with the HMAC MAC key type.
Key type (one required)	
CIPHER	Specifies that this key is for data-encryption. Only valid for AES algorithm. See Figure 1 on page 29 and Table 18 on page 47 for key-usage and key-management keywords.
DKYGENKY	Specifies that this key is for key-generation. Only valid for AES algorithm. See Figure 6 on page 39 and Table 18 on page 47 for key-usage and key-management keywords.
EXPORTER	Specifies that this key is an EXPORTER key-encrypting key. Only valid for AES algorithm. See Figure 4 on page 35 and Table 18 on page 47 for key-usage and key-management keywords.
IMPORTER	Specifies that this key is an IMPORTER key-encrypting key. Only valid for AES algorithm. See Figure 5 on page 37 and Table 18 on page 47 for key-usage and key-management keywords.
MAC	Specifies that this key is for message authentication code operations. Valid for HMAC and AES algorithms. See Figure 2 on page 31, Figure 3 on page 33, and Table 18 on page 47 for key-usage and key-management keywords.
PINCALC	Specifies that this key is for calculating PINs. Only valid for AES algorithm. See Figure 7 on page 42 and Table 18 on page 47 for key-usage and key-management keywords.
PINPROT	Specifies that this key is for wrapping and unwrapping PIN blocks. Only valid for AES algorithm. See Figure 8 on page 44 and Table 18 on page 47 for key-usage and key-management keywords.
PINPRW	Specifies that this key is for generating and verifying PIN reference values. Only valid for AES algorithm. See Figure 9 on page 46 and Table 18 on page 47 for key-usage and key-management keywords.

clear_key_bit_length

Direction	Type
Input	Integer

The length of the clear key in bits. Specify 0 when no key value is supplied (Key status rule NO-KEY). Specify a valid key bit length when a key value is supplied (Key status rule KEY-CLR):

- For HMAC algorithm, MAC key type, this is a value between 80 and 2048.
- For AES algorithm, CIPHER/EXPORTER/IMPORTER key types, this is a value of 128, 192, or 256.

clear_key_value

Direction	Type
Input	String

This parameter is used when the KEY-CLR keyword is specified. This parameter is the clear key value to be put into the token being built.

key_name_length

Direction	Type
Input	Integer

The length of the *key_name* parameter. Valid values are 0 and 64.

key_name

Direction	Type
Input	String

A 64-byte key store label to be stored in the associated data structure of the token.

user_associated_data_length

Direction	Type
Input	Integer

The length of the user-associated data. The valid values are 0 to 255 bytes.

user_associated_data

Direction	Type
Input	String

User-associated data to be stored in the associated data structure.

token_data_length

Direction	Type
Input	Integer

This parameter is reserved. The value must be 0.

token_data

Direction	Type
Ignored	Integer

This parameter is ignored.

service_data_length

Direction	Type
Input	Integer

The length of the *service_data* parameters in bytes. For rule array keyword DKYUSAGE, the value must be a multiple of 8. Otherwise, the value must be 0. The maximum value is 280.

service_data

Direction	Type
Input	String

Data to be processed by this service when building the skeleton token. If the DKYUSAGE keyword is specified in the rule array, this parameter contains an array of key usage keywords for the type of key to be derived. The keywords are 8 bytes in length and must be left-aligned and padded on the right with blanks.

target_key_token_length

Direction	Type
Input/Output	Integer

On input, the length of the *target_key_token* parameter supplied to receive the token. On output, the actual length of the token returned to the caller. Maximum length is 725 bytes.

target_key_token

Direction	Type
Output	String

The key token built by this service.

Usage Notes

The topic contains information for all key types detailing the key-usage and key-management keywords that are supported for each key type.

Figure 1 on page 29 shows all the valid keyword combinations and their defaults for AES key type CIPHER. For a description of these keywords, see Table 18 on page 47.

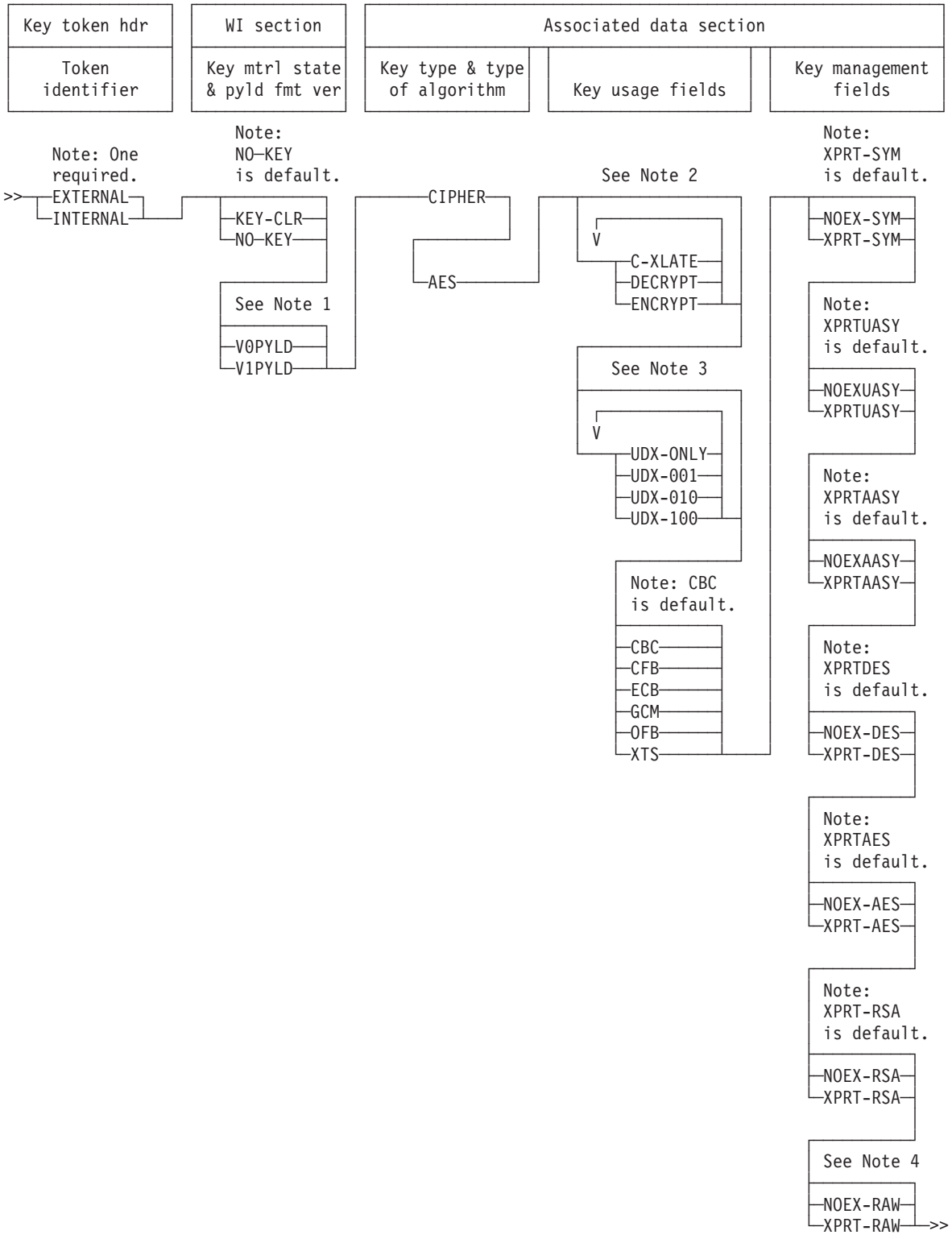

Figure 1. Key Token Build2 keyword combinations for AES CIPHER keys

Note:

1. Keyword V0PYLD is the default for compatibility reasons. V1PYLD is recommended.
2. Keywords DECRYPT and ENCRYPT are defaults unless one or more keywords in the group are specified.

3. Choose any number of keywords in this group. No keywords in the group are defaults.
4. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Figure 2 on page 31 shows all the valid keyword combinations and their defaults for AES key type MAC. For a description of these keywords, see Table 18 on page 47.

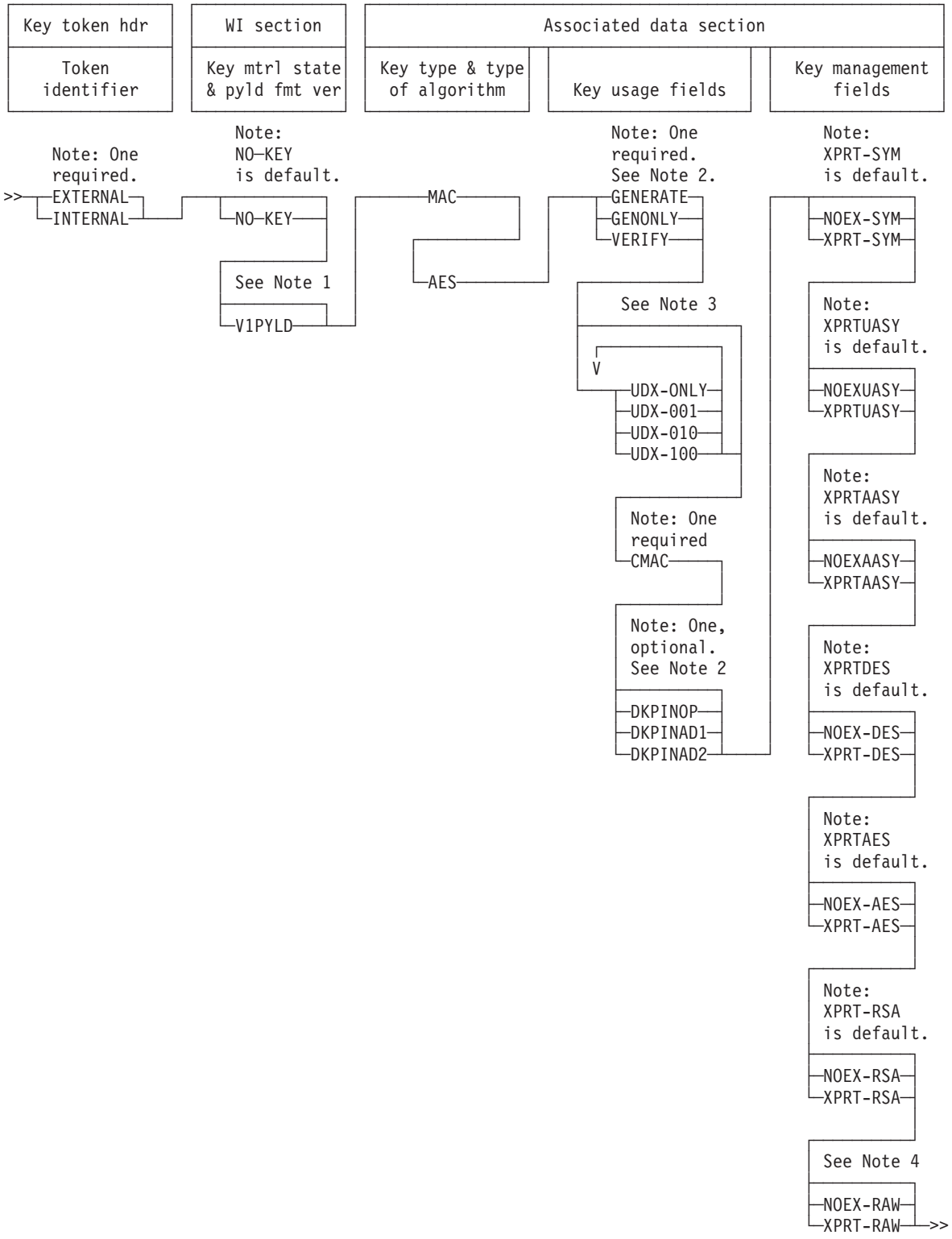

Figure 2. Key Token Build2 keyword combinations for AES MAC keys

Note:

1. Keyword V1PYLD is the default. V1PYLD is the only payload format version allowed for this key type.
2. Keyword GENERATE is not valid with keywords DKPINOP, DKPINAD1, and DKPINAD2.

3. Choose any number of keywords in this group. No keywords in the group are defaults.
4. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Figure 3 on page 33 shows all the valid keyword combinations and their defaults for HMAC key type MAC. For a description of these keywords, see Table 18 on page 47.

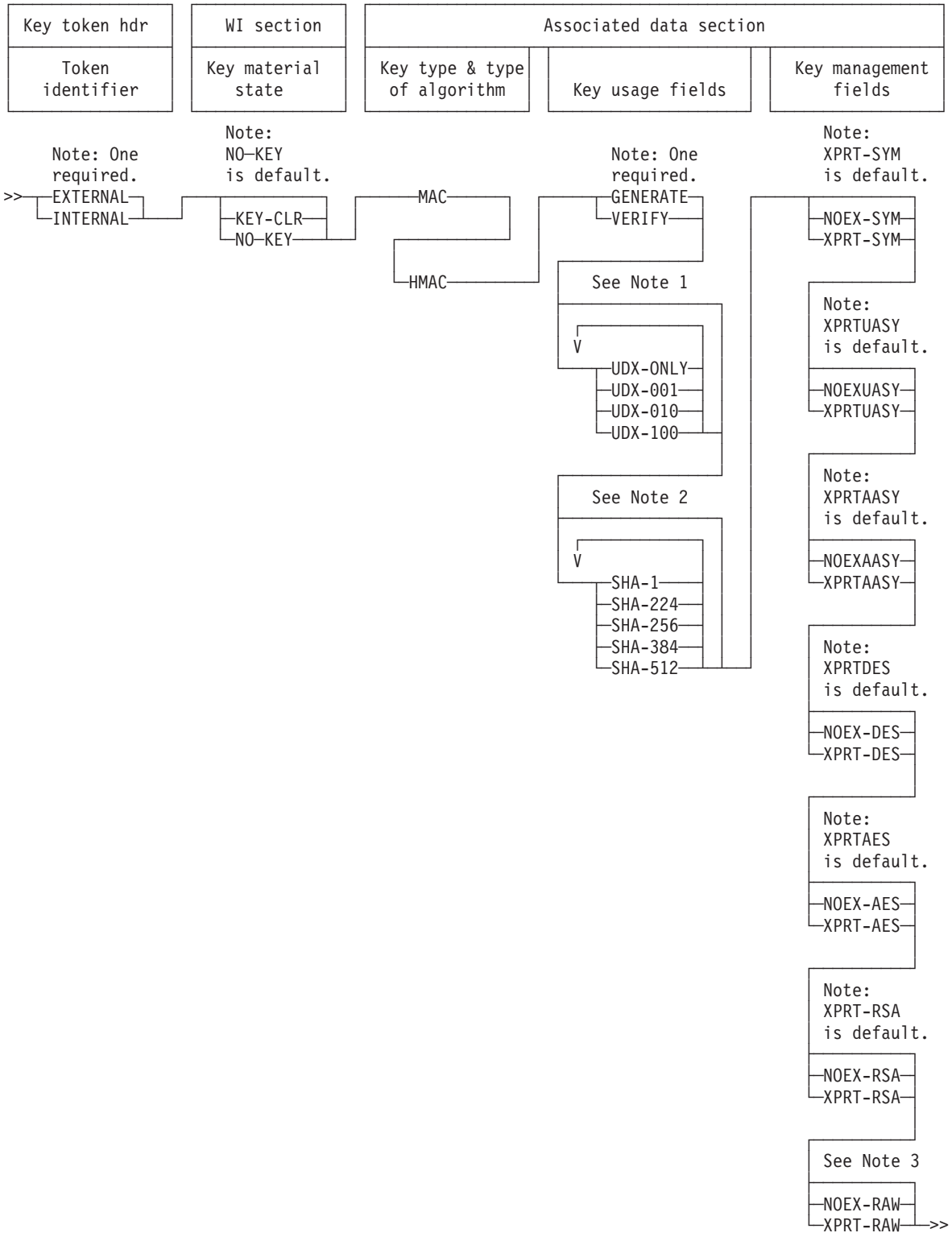

Figure 3. Key_Token_Build2 keyword combinations for HMAC MAC keys

Note:

1. Choose any number of keywords in this group. No keywords in the group are defaults.
2. All keywords in the group are defaults unless one or more keywords in the group are specified.

3. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Figure 4 on page 35 shows all the valid keyword combinations and their defaults for AES key type EXPORTER. For a description of these keywords, see Table 18 on page 47.

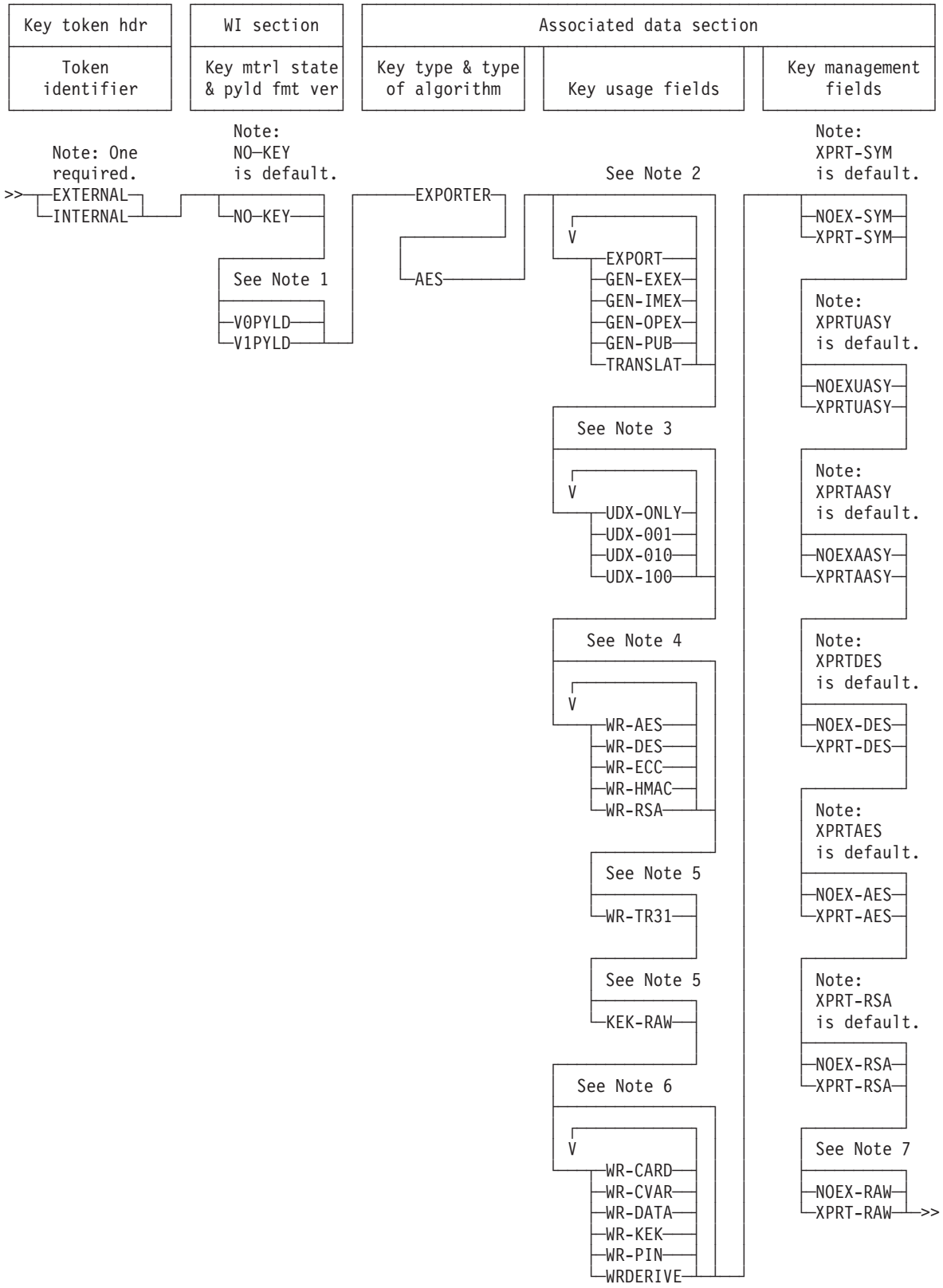

Figure 4. Key Token Build2 keyword combinations for AES EXPORTER keys

Note:

1. Keyword V0PYLD is the default for compatibility reasons. V1PYLD is recommended.
2. All keywords in the group are defaults unless one or more keywords in the group are specified.
3. Choose any number of keywords in this group. No keywords in the group are defaults.
4. Keywords WR-AES, WR-DES, and WR-HMAC are defaults unless one or more keywords in the group are specified.
5. This keyword is for future use and its meaning is currently undefined.
6. Keywords WR-CARD, WR-DATA, WR-KEK, WR-PIN, and WRDERIVE in the group are defaults unless one or more keywords in the group are specified. 2.
7. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Figure 5 on page 37 shows all the valid keyword combinations and their defaults for AES key type IMPORTER. For a description of these keywords, see Table 18 on page 47.

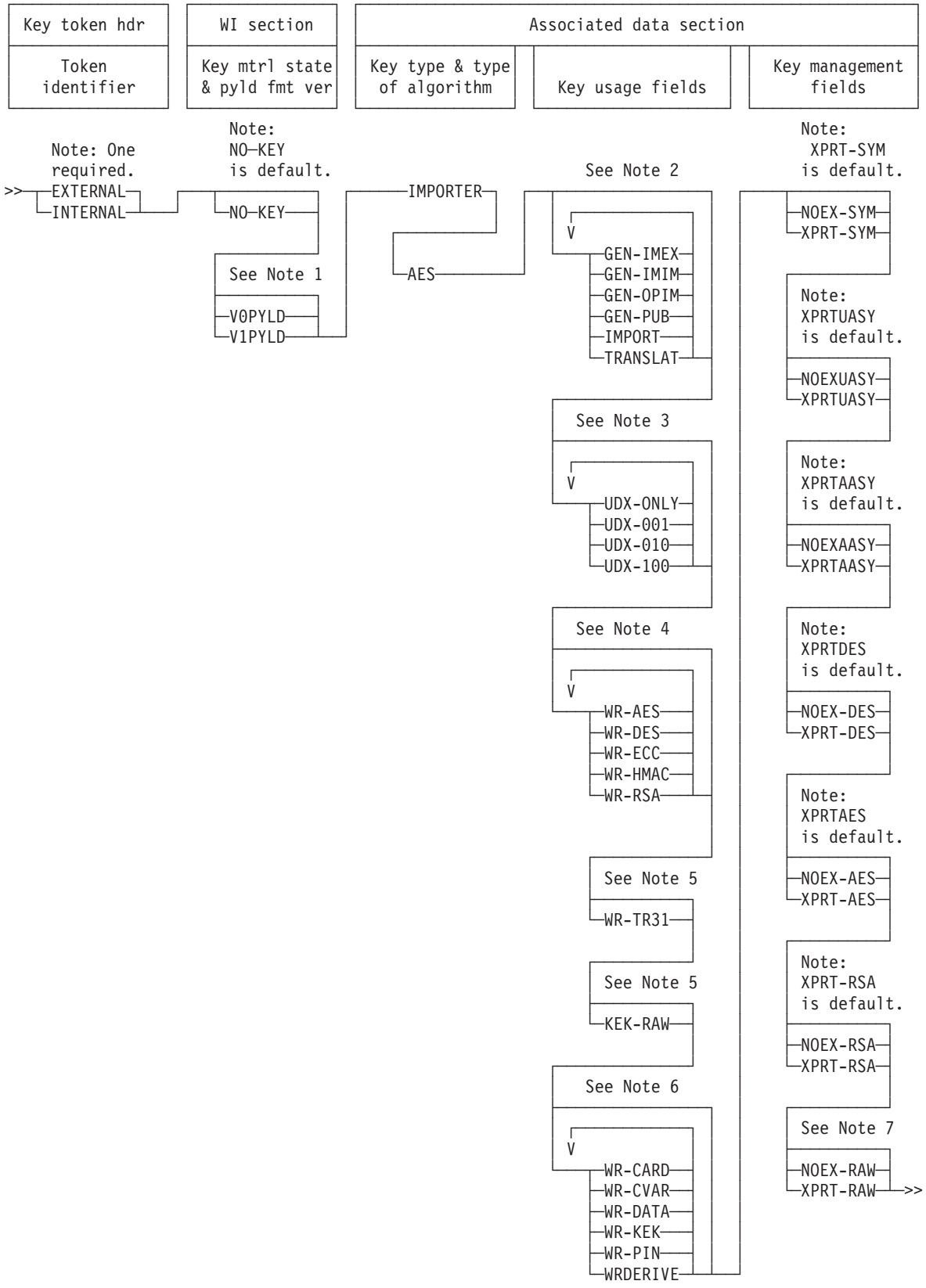

Figure 5. Key Token Build2 keyword combinations for AES IMPORTER keys

Note:

1. Keyword V0PYLD is the default for compatibility reasons. V1PYLD is recommended.
2. All keywords in the group are defaults unless one or more keywords in the group are specified.
3. Choose any number of keywords in this group. No keywords in the group are defaults. 4.
4. Keywords WR-AES, WR-DES, and WR-HMAC are defaults unless one or more keywords in the group are specified.
5. This keyword is for future use and its meaning is currently undefined.
6. Keywords WR-CARD, WR-DATA, WR-KEK, WR-PIN, and WRDERIVE in the group are defaults unless one or more keywords in the group are specified.
7. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Figure 6 on page 39 shows all the valid keyword combinations and their defaults for AES key type DKYGENKY. For a description of these keywords, see Table 18 on page 47.

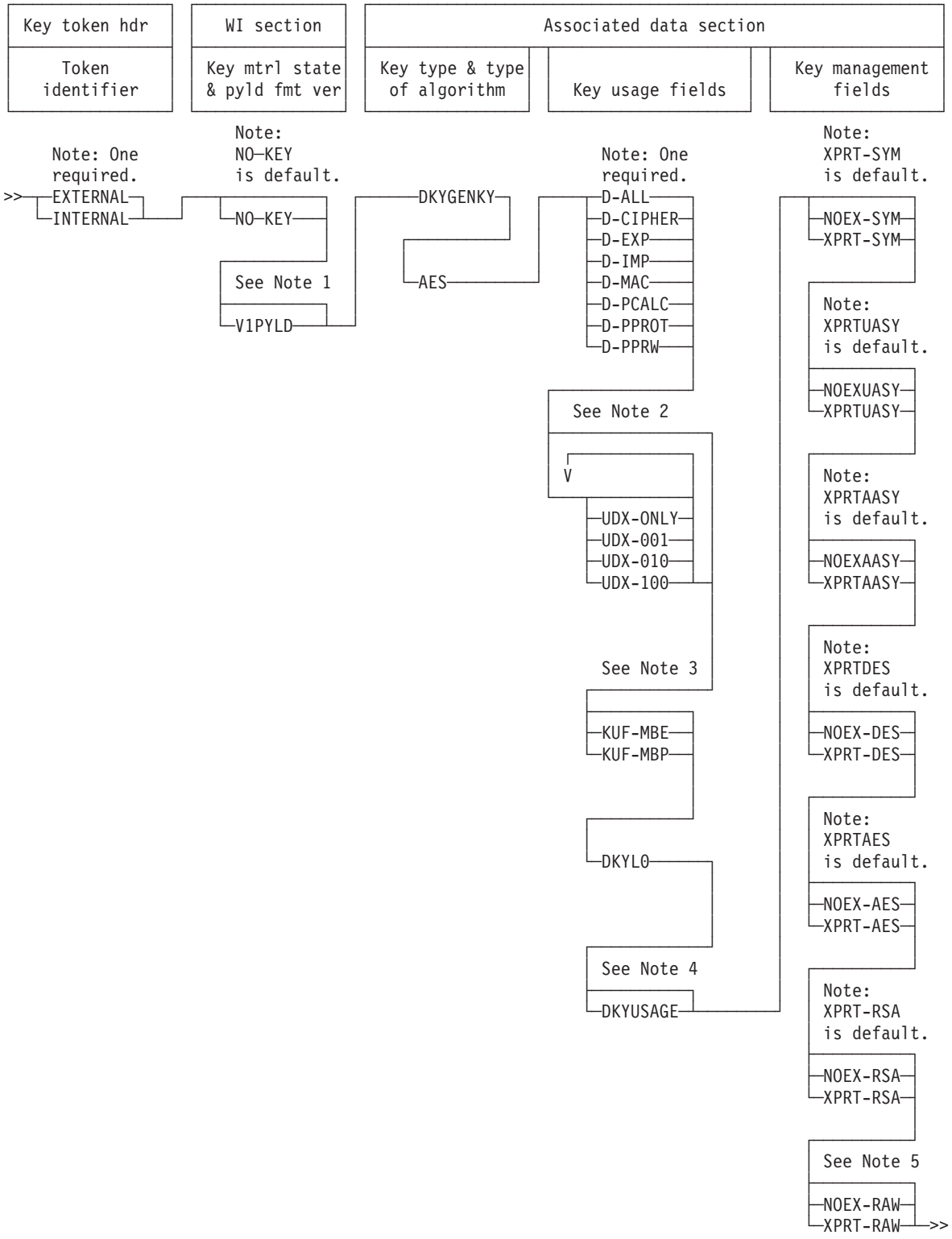

Figure 6. Key Token Build2 keyword combinations for AES DKYGENKY keys

Note:

1. Keyword V1PYLD is the default. V1PYLD is the only payload format version allowed for this key type.
2. Choose any number of keywords in this group.

3. If keyword D-ALL is specified, keywords KUF-MBE and KUF-MBP are not valid and there is no default; otherwise the default is KUF-MBE. If keyword DKPINOP, DKPINOPP, DKPINAD1, or DKPINAD2 is specified in the *service_data* parameter (in other words, the type of key to diversify is DK enabled), keyword KUF-MBP is not valid.
4. There is no default. DKYUSAGE specifies that the *service_data* parameter contains key-usage field keywords related to the type of key to diversify. These related attributes become part of the key usage fields of the DKYGENKY diversifying key. They are related because they are used to control which key usage attributes are permissible in the generated diversified key. To generate a diversified key, use the Diversified_Key_Generate2 verb. The type of key to diversify affects the use of the DKYUSAGE keyword:
   - For D-ALL, DKYUSAGE is not allowed.
   - For D-CIPHER, D-EXP, and D-IMP, DKYUSAGE is optional. If not specified, default key usage fields of the type of key to diversify are used.
   - For D-MAC, D-PCALC, D-PPROT, and D-PPRW, DKYUSAGE is required.

   See Table 17 for additional information on the meaning of the verb_data variable when DKYUSAGE is specified.
5. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Table 17. Meaning of service_data parameter when DKYUSAGE specified

Type of key to diversify	DKYUSAGE keyword	Description of verb_data variable when DKYUSAGE specified
D-ALL	Not allowed	Not applicable.
D-CIPHER	Optional	If keyword DKYUSAGE is specified, the service_data parameter must contain key usage fields keywords related to an AES CIPHER key. If not specified, the related key usage fields will be that of a default AES CIPHER key.
D-EXP	Optional	If keyword DKYUSAGE is specified, the service_data parameter must contain key usage fields keywords related to an AES EXPORTER key. If not specified, the related key usage fields will be that of a default AES EXPORTER key.
D-IMP	Optional	If keyword DKYUSAGE is specified, the service_data parameter must contain key usage fields keywords related to an AES IMPORTER key. If not specified, the related key usage fields will be that of a default AES IMPORTER key.
D-MAC	Required	The service_data parameter must contain key usage fields keywords related to an AES MAC key.
D-PCALC	Required	The service_data parameter must contain key usage fields keywords related to an AES PINCALC key.
D-PPROT	Required	The service_data parameter must contain key usage fields keywords related to an AES PINPROT key.

Table 17. Meaning of service_data parameter when DKYUSAGE specified (continued)

Type of key to diversify	DKYUSAGE keyword	Description of verb_data variable when DKYUSAGE specified
D-PPRW	Required	The service_data parameter must contain key usage fields keywords related to an AES PINPRW key.

Figure 7 on page 42 shows all the valid keyword combinations and their defaults for AES key type PINCALC. For a description of these keywords, see Table 18 on page 47.

Key Token Build2

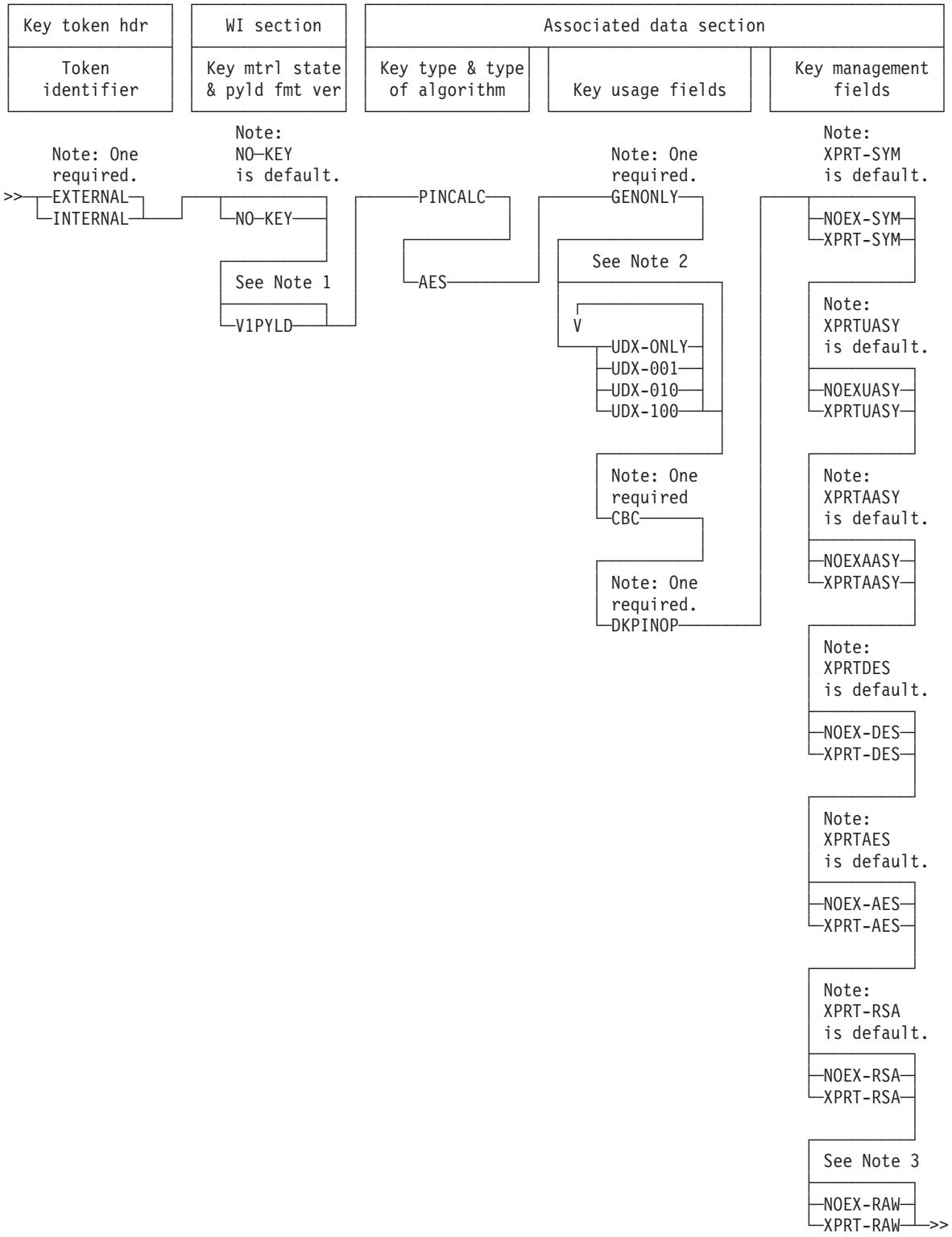

Figure 7. Key Token Build2 keyword combinations for AES PINCALC keys

Note:

1. Keyword V1PYLD is the default. V1PYLD is the only payload format version allowed for this key type.
2. Choose any number of keywords in this group. No keywords in the group are defaults.

3. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Figure 8 on page 44 shows all the valid keyword combinations and their defaults for AES key type PINPROT. For a description of these keywords, see Table 18 on page 47.

Key Token Build2

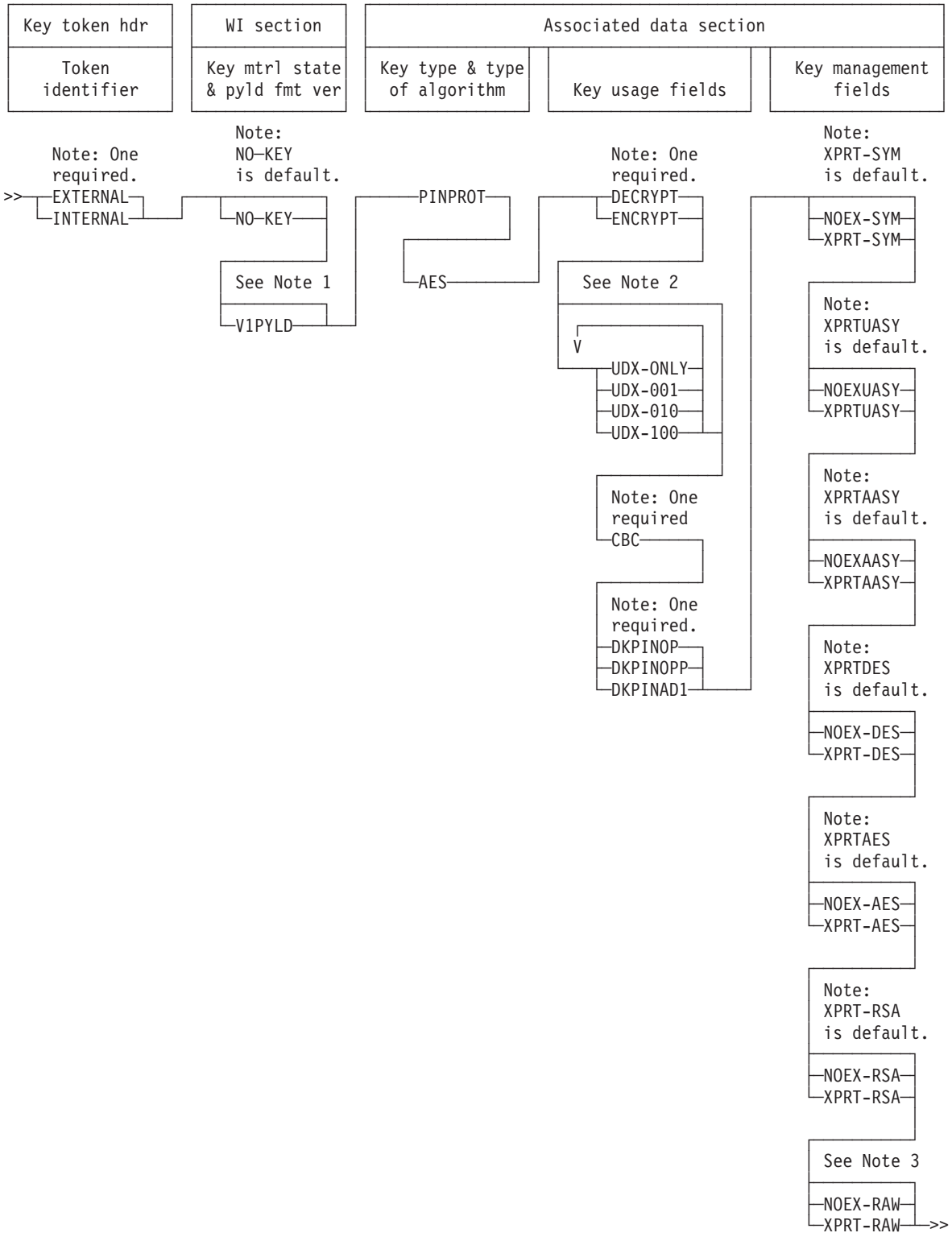

Figure 8. Key Token Build2 keyword combinations for AES PINPROT keys

Note:

1. Keyword V1PYLD is the default. V1PYLD is the only payload format version allowed for this key type.
2. Choose any number of keywords in this group. No keywords in the group are defaults.

3. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Figure 9 on page 46 shows all the valid keyword combinations and their defaults for AES key type PINPRW. For a description of these keywords, see Table 18 on page 47.

Key Token Build2

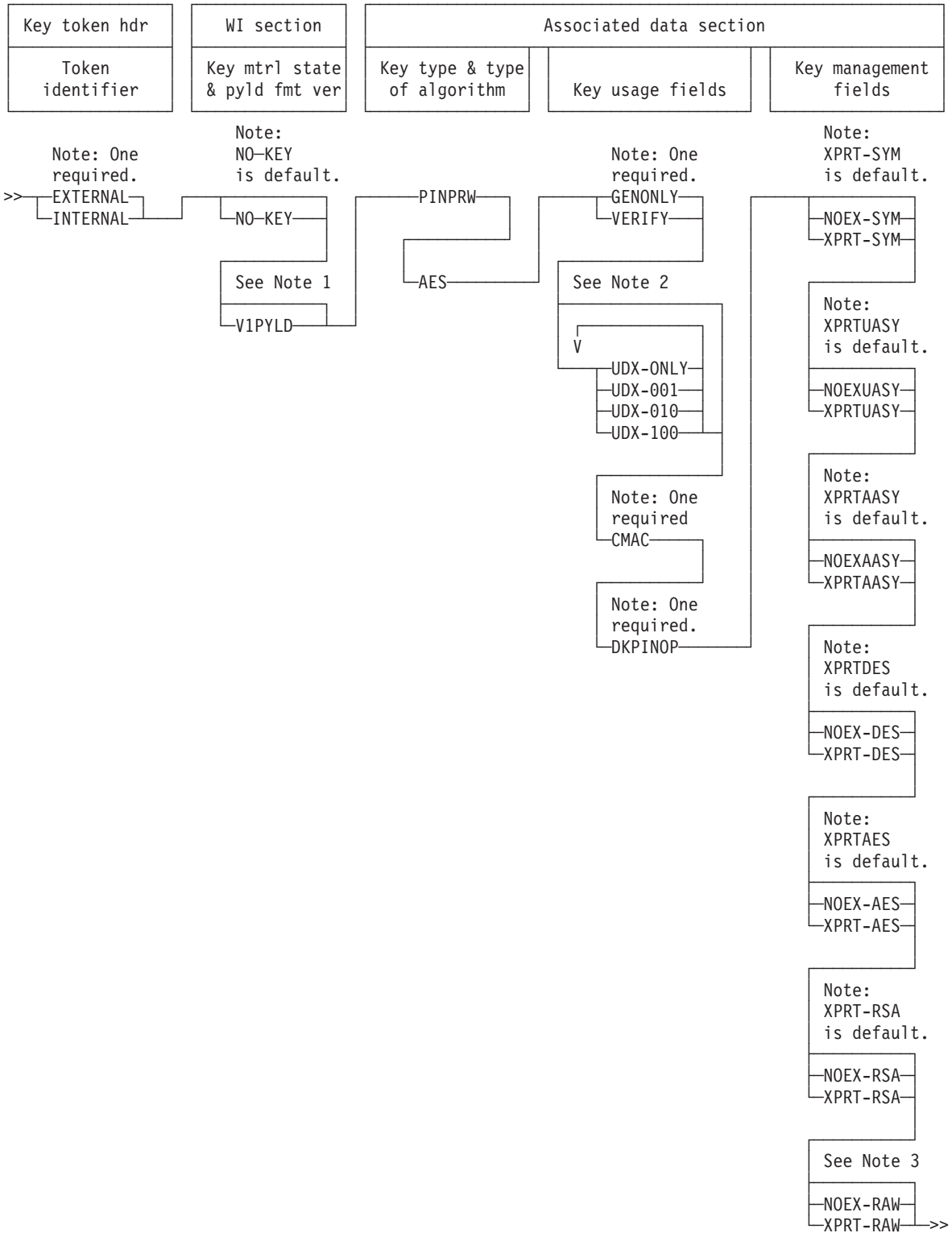

Figure 9. Key Token Build2 keyword combinations for AES PINPRW keys

Note:

1. Keyword V1PYLD is the default. V1PYLD is the only payload format version allowed for this key type.
2. Choose any number of keywords in this group. No keywords in the group are defaults.

3. NOEX-RAW is default. These keywords are for future use and their meanings are currently undefined.

Table 18. Key Token Build2 key-usage keywords

Keyword	Meaning
Key-usage field 1, high-order byte	
Generate control (one required). ***For MAC key type. For AES PINCALC and PINPRW key types.***	
GENERATE	Specifies that this key can be used to generate a MAC. A key that can generate a MAC can also verify a MAC.
GENONLY	Specifies that this key can only be used to generate a MAC. It can not be used to verify a MAC or PRW. Valid only for AES MAC, PINCALC, and PINPRW key types.
VERIFY	Specifies that this key cannot be used to generate a MAC. It can only be used to verify a MAC.
Encrypt control. ***For AES algorithm, CIPHER key types (optional, any combination).*** **Note:** All keywords in the list below are defaults unless one or more keywords in the list are specified. ***For AES algorithm, PINPROT key types (one, required).***	
ENCRYPT	Specifies that this key can be used to encipher data.
DECRYPT	Specifies that this key can be used to decipher data.
Ciphertext Translate Control (optional). ***For AES algorithm, CIPHER key type.***	
C-XLATE	Specifies that this key can only be used for cipher text translation.
Exporter control (any combination, optional). ***For AES algorithm, EXPORTER key type.*** **Note:** All keywords in the list below are defaults unless one or more keywords in the list are specified.	
EXPORT	Specifies that this key can be used for export.
TRANSLAT	Specifies that this key can be used for translate.
GEN-OPEX	Specifies that this key can be used for generate OPEX.
GEN-IMEX	Specifies that this key can be used for generate IMEX.
GEN-EXEX	Specifies that this key can be used for generate EXEX.
GEN-PUB	Specifies that this key can be used for generate PUB.
Importer control (any combination, optional). ***For AES algorithm, IMPORTER key type.*** **Note:** All keywords in the list below are defaults unless one or more keywords in the list are specified.	
IMPORT	Specifies that this key can be used for import.
TRANSLAT	Specifies that this key can be used for translate.
GEN-OPIM	Specifies that this key can be used for generate OPIM.
GEN-IMEX	Specifies that this key can be used for generate IMEX.
GEN-IMIM	Specifies that this key can be used for generate IMIM.
GEN-PUB	Specifies that this key can be used for generate PUB.
Key derivation control (one, required). ***For AES algorithm, DKYGENKY key type.***	
D-ALL	Specifies that this can derive any AES key type listed in this section.
D-CIPHER	Specifies that this key can derive an AES CIPHER key.
D-EXP	Specifies that this key can derive an AES EXPORTER key.
D-IMP	Specifies that this key can derive an AES IMPORTER key.

Table 18. Key Token Build2 key-usage keywords (continued)

Keyword	Meaning
D-MAC	Specifies that this key can derive an AES MAC key.
D-PCALC	Specifies that this key can derive an AES PINCALC key.
D-PPROT	Specifies that this key can derive an AES PINPROT key.
D-PPRW	Specifies that this key can derive an AES PINPRW key
Key-usage field 1, low-order byte	
User-defined extension control (any combination, optional). ***For all algorithms and key types.*** **Note:** The default is such that the key can be used in both UDXs and CCA and none of the user-defined UDX bits are set.	
UDX-ONLY	Specifies that this key can only be used in UDXs.
UDX-001	Specifies that the rightmost user-defined UDX bit is set.
UDX-010	Specifies that the middle user-defined UDX bit is set.
UDX-100	Specifies that the leftmost user-defined UDX bit is set.
Key-usage field 2, high-order byte	
Hash method control (any combination, optional). ***For HMAC algorithm, MAC key type.*** **Note:** All keywords in the list below are defaults unless one or more keywords in the list are specified.	
SHA-1	Specifies that the SHA-1 hash method is allowed for the key.
SHA-224	Specifies that the SHA-224 hash method is allowed for the key.
SHA-256	Specifies that the SHA-256 hash method is allowed for the key.
SHA-384	Specifies that the SHA-384 hash method is allowed for the key.
SHA-512	Specifies that the SHA-512 hash method is allowed for the key.
Mode control (one, optional). ***For AES algorithm, CIPHER key type.***	
CBC	Specifies that this key can be used for cipher block chaining. This is the default.
CFB	Specifies that this key can be used for cipher feedback.
ECB	Specifies that this key can be used for electronic code book.
GCM	Specifies that this key can be used for Galois/counter mode.
OFB	Specifies that this key can be used for output feedback.
XTS	Specifies that this key can be used for Xor-Encrypt-Xor-based Tweaked Stealing.
Mode control (one, required). ***For AES algorithm, MAC or PINPRW key types.***	
CMAC	MAC calculation mode is block cipher-based MAC algorithm.
Mode control (one, required). ***For AES algorithm, PINPROT or PINCALC key type.***	
CBC	Specifies that this key can be used for cipher block chaining.
Key-usage field level of control (one, optional). ***For AES algorithm, DKYGENKY key type.*** **Note:** Not valid when D-ALL key derivation control is specified.	
KUF-MBE	Specifies that the key usage fields of the key to be generated must be equal to the related generated key usage fields of the DKYGENKY generating key. This is the default.

Table 18. Key Token Build2 key-usage keywords (continued)

Keyword	Meaning
KUF-MBP	Specifies that the key usage fields of the key to be generated must be permitted based on the related generated key usage fields of the DKYGENKY generating key. The key to be diversified is not permitted to have a higher level of usage than the related key usage fields permit. The key to be diversified is only permitted to have key usage that is less than or equal to the related key usage fields. The UDX-ONLY bit of the related key usage fields must always be equal in both the generating key and the generated key. Note: This value is not valid if the key is to be used to derive keys for the DK PIN methods.
Key-usage field 2, low-order byte	
Key-encrypting key control (any combination, optional). ***For AES algorithm, EXPORTER or IMPORTER key type.*** **Note:** The default is such that the key cannot export a RAW key nor wrap or unwrap a TR-31 key block.	
KEK-RAW	Specifies that this key-encrypting key can export a RAW key. A RAW key is a key that is encrypted, but does not have any associated data.
WR-TR31	Specifies that this key-encrypting key can wrap or unwrap a TR-31 key block.
Key-derivation sequence level (one, required). ***For AES algorithm, DKYGENKY key type.***	
DKYL0	Specifies that this key-generating key can be used to derive the key specified by the Key derivation and Derived key usage controls.
Key-usage field 3, high-order byte	
Key-usage wrap algorithm control (any combination, optional). ***For AES algorithm, EXPORTER or IMPORTER key type.*** **Note:** Keywords WR-DES, WR-AES, and WR-HMAC are defaults unless one or more keywords are specified.	
WR-DES	Specifies that this key can be used to wrap DES keys.
WR-AES	Specifies that this key can be used to wrap AES keys.
WR-HMAC	Specifies that this key can be used to wrap HMAC keys.
WR-RSA	Specifies that this key can be used to wrap RSA keys.
WR-ECC	Specifies that this key can be used to wrap ECC keys.
German Banking Industry Committee PIN method Command key-usage. ***For AES algorithm, MAC key type (One, optional).*** ***For AES algorithm, PINCALC, PINPROT, or PINPRW key types (One, required).*** **Note:** • One keyword is required for keys to be used with the DK PIN services. • The command keyword required depends on the service being called. See the description of the key identifiers of the service for the required command key-usage flag.	
DKPINAD1	Specifies that this key may be used to create or verify a pin block to allow changing the account number associate with a PIN. Valid only with AES PINPROT or MAC key types.
DKPINAD2	Specifies that this key may be used to create or verify an account change string to allow changing the account number associated with a PIN. Valid only with AES MAC.
DKPINOP	Specifies that this key may be used as a general-purpose key. It may not be used as a special-purpose key. Valid for AES MAC, PINCALC, PINPROT, or PINPRW key types.
DKPINOPP	Specifies that this key is to be used to encrypt a PBF-1 format pin block for the specific purpose of creating a PIN mailer. Valid only with AES PINPROT key type with the ENCRYPT keyword specified.

Table 18. Key Token Build2 key-usage keywords (continued)

Keyword	Meaning
Verb data content (required for D-MAC, D-PPROT, D-PCALC and D-DPRW; otherwise optional). ***For AES algorithm, DKYGENKY key type.*** **Note:** Not valid when D-ALL key derivation control is specified.	
DKYUSAGE	Specifies that the service_data parameter identifies key usage information for a DKYGENKY. This information pertains to the allowable key usage of the key to be derived.
Key-usage field 3, low-order byte	
Byte is reserved (except for DK-enabled keys AES MAC, PINCALC, PINPROC, PINPRW).	
Key-usage field 4, high-order byte	
Key-usage wrap class control (any combination, optional). ***For AES algorithm, EXPORTER or IMPORTER key type.*** **Note:** Keywords WR-DATA, WR-KEK, WR-PIN, WRDERIVE and WR-CARD in the list below are defaults unless one or more keywords in the list are specified.	
WR-DATA	Specifies that this key can be used to wrap DATA class keys.
WR-KEK	Specifies that this key can be used to wrap KEK class keys.
WR-PIN	Specifies that this key can be used to wrap PIN class keys.
WRDERIVE	Specifies that this key can be used to wrap DERIVATION class keys.
WR-CARD	Specifies that this key can be used to wrap CARD class keys.
WR-CVAR	Specifies that this key can be used to wrap CVAR class keys.
Key-usage field 4, low-order byte	
This byte is reserved.	
Key-management field 1, high-order byte	
Symmetric-key export control (one, optional). ***Key-management field 1 for all algorithms and key types.***	
NOEX-SYM	Prohibits the export of the key with a symmetric key.
XPRT-SYM	Permits the export of the key with a symmetric key. This is the default.
Unauthenticated asymmetric-key export control (one, optional). ***Key-management field 1 for all algorithms and key types.***	
NOEXUASY	Prohibits the export of the key with an unauthenticated asymmetric key.
XPRTUASY	Permits the export of the key with an unauthenticated asymmetric key. This is the default.
Authenticated asymmetric-key export control (one, optional). ***Key-management field 1 for all algorithms and key types.***	
NOEXAASY	Prohibits the export of the key with an authenticated asymmetric key.
XPRTAASY	Permits the export of the key with an authenticated asymmetric key. This is the default.
Key-management field 1, low-order byte	
RAW-format export control (one, optional). ***Key-management field 1 for all algorithms and key types.***	
NOEX-RAW	Prohibits the export of the key in RAW format. This is the default.
XPRT-RAW	Permits the export of the key in RAW format.
DES-key export control (one, optional). ***Key-management field 1 for all algorithms, all key types.***	
NOEX-DES	Prohibits the export of the key using DES key.

Table 18. Key Token Build2 key-usage keywords (continued)

Keyword	Meaning
XPRT-DES	Permits the export of the key using DES key. This is the default.
AES-key export control (one, optional). ***Key-management field 1 for all algorithms, all key types.***	
NOEX-AES	Prohibits the export of the key using AES key.
XPRT-AES	Permits the export of the key using AES key. This is the default.
RSA-key export control (one, optional). ***Key-management field 1 for all algorithms, all key types.***	
NOEX-RSA	Prohibits the export of the key using RSA key.
XPRT-RSA	Permits the export of the key using RSA key. This is the default.
Key-management field 2, high-order byte	
Byte contains key completeness. There is no user-defined content.	
Key-management field 2, low-order byte	
Byte contains security history. There is no user-defined content.	
Key-management field 3, high-order byte	
Byte contains pedigree original rules. There is no user-defined content.	
Key-management field 3, low-order byte	
Byte contains pedigree current rules. There is no user-defined content.	

Building a DKYGENKY Key: The way that the DKYGENKY tokens are built is different from the way they were previously built. The token layout itself has been updated. The DKYGENKY key is used to derive other key types.

In order to control the key usage of the key to be derived, key usage field information for the derived key is included in the DKYGENKY token. Consider these scenarios based on the type of key to derive:

- DKYGENKY has a type of key to derive of D-ALL.

  This type of key is allowed to derive any of the allowed key types. No key usage field information is included in this key. Usage is determined by the skeleton token identified by the generated_key_identifier parameter of the CSNBDKG2 callable service. A special access control point must be enabled in the active role to use this option.
- DKYGENKY has a type of key to be derived that has default key usage (D-CIPHER, D-EXP, D-IMP).

  Several key types have default key usage defined, while other key types do not. For those key types which have default key usage defined (D-CIPHER, D-EXP, and D-IMP), the only requirement is to specify the type of key to derive. The default key usage fields is included in the DKYGENKY key, beginning with key usage field 3.
- DKYGENKY has a type of key to be derived that requires non-default key usage.

  If non-default key usage of a key to be derived is required or desired, specify rule array keyword DKYUSAGE. With this keyword, the verb_data parameter is used to identify all of the key-usage field keywords for the key to be diversified. Do not specify any token identifier, type of algorithm, key type, or key management field keywords. Set the verb_data_length value to the number of bytes in the verb_data variable. This length must be a multiple of 8.

When rule array keyword DKYUSAGE is specified, choose between whether the key usage field attributes in the DKYGENKY starting at key-usage field 3 have the strictest control (KUF-MBE or 'must be equal', which is the default) or allow flexibility in the key usage attributes of the key to be generated (KUF-MBP or 'must be permitted').

Choosing KUF-MBE ('key usage fields must be equal') provides a one-to-one mapping of usage fields between the generating key and the generated key. The key usage fields related to the key to be diversified in the DKYGENKY key must match exactly with the key usage fields of any skeleton key provided as input to the CSNBDKG2 callable service.

Choosing KUF-MBP ('key usage fields must be permitted') provides that the key to be diversified is allowed to have any key usage attribute that is enabled in the DKYGENKY. For example, if a DKYGENKY with D-EXP usage has default EXPORTER key usage fields, KUF-MBP allows the diversified EXPORTER key to have only the EXPORT bit on in key-usage field 1. This is permitted because the diversified key actually is more restrictive than the usage allowed by the DKYGENKY key. Conversely, if a DKYGENKY with D-EXP usage has only the EXPORT bit on in key-usage field 3 (which maps to key usage field 1 of the diversified EXPORTER key), it would not be permitted for the skeleton key used as input to the CSNBDKG2 callable service to have the XLATE bit on in key-usage field 1.

Note:

- For rule array keyword KUF-MBP, one exception exists where the value of the UDX-ONLY bit in key usage field 3 of a DKYGENKY key must always match the value of the UDX-ONLY bit in key usage field 1 of the diversified key.
- Under access control point control, there is one case where a many-to-one mapping is permitted and verb data is not used. This case is when you specify D-ALL which says any allowable key type can be derived.

Required Hardware

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

Table 19. Key Token Build2 required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890	None	
IBM System z9 EC IBM System z9 BC	None	
IBM System z10 EC IBM System z10 BC	None	
IBM zEnterprise 196 IBM zEnterprise 114	None	
IBM zEnterprise EC12 IBM zEnterprise BC12	None	

Key Translate2 (CSNBKTR2 and CSNEKTR2)

The Key Translate2 callable service translates the *input_key_token* parameter in one of several ways:

- Changes an external DES or variable-length symmetric key token from encipherment under one key-encrypting key to another.
- Changes the wrapping method of an external DES key token.
- Converts an operational AES DATA token (version X'04') to an operational AES CIPHER token (version X'05') or converts an operational AES CIPHER token (version X'05') to an operational AES DATA token (version X'04').
- Converts a key token using the AESKW wrapping method from a variable length payload to a fixed length payload.

To reencipher a key token, specify the TRANSLAT rule array keyword (the default), the external key token, and the input and output key-encrypting keys. If the *input_key_token* is a DES key token, you can also specify which key wrapping method to use. If no wrapping method is specified, the system default wrapping method will be used.

To change the wrapping method of an external DES key token, specify the REFORMAT rule array keyword, the Key Wrapping Method to use, the external key token and the input key-encrypting key. If no wrapping method is specified, the system default wrapping method will be used. Note that the *output_KEK_identifier* will be ignored.

To convert an operational AES DATA token (version X'04') to an operational AES CIPHER token (version X'05') or vice versa, specify the REFORMAT rule array keyword, the operational key token as *input_key_token*, and either a NULL token or skeleton token as *output_key_token*. Note that both the *input_KEK_identifier* and the *output_KEK_identifier* will be ignored as the corresponding lengths must be zero.

To convert an internal or external variable-length AES key token (version X'05') from a variable-length payload to a fixed-length payload, specify the V1PYLD rule array keyword. The fixed-length payload will obfuscate the key length. This keyword is only valid for the CIPHER, EXPORTER and IMPORTER key types.

To convert an internal or external variable-length AES key token (version X'05') from a fixed-length payload to a variable-length payload, specify the V0PYLD rule array keyword. This keyword is only valid for the CIPHER, EXPORTER and IMPORTER key types.

Note: All key labels must be unique.

Format

```
CALL CSNBKTR2(
              return_code,
              reason_code,
              exit_data_length,
              exit_data,
              rule_array_count,
              rule_array,
              input_key_length,
              input_key_token,
              input_KEK_length,
              input_KEK_identifier,
              output_KEK_length,
              output_KEK_identifier,
              output_key_length,
              output_key_token )
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. “ICSF and TSS Return and Reason Codes” on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. “ICSF and TSS Return and Reason Codes” on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFFFF' (2 gigabytes). The data is defined 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 count must be between 0 and 4, inclusive.

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service. 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.

Keyword	Meaning
Encipherment (optional)	

Keyword	Meaning
REFORMAT	Reformat the input_key_token. • When *input_key_token* is a DES key token, reformat with the Key Wrapping Method specified. • When *input_key_token* is an operational AES key token, either reformat an AES DATA key (version X'04') to an AES CIPHER key (version X'05') or the reverse (version X'05' to version X'04').
TRANSLAT	Translate the *input_key_token* from encipherment under the *input_KEK_identifier* to encipherment under the *output_KEK_identifier*. This is the default.
V1PYLD	Reencipher an input variable-length AES key token (version X'05') to a payload version1 (fixed-length) key token. This keyword is only valid for the CIPHER, EXPORTER and IMPORTER key types.
V0PYLD	Reencipher an input variable-length AES key token (version X'05') to a payload version 0 (variable-length) key token. This keyword is only valid for the CIPHER, EXPORTER and IMPORTER key types.
Key Wrapping Method (optional, valid only if** input_key_token **is an external DES key token)	
USECONFG	Specifies that the system default configuration should be used to determine the wrapping method. This is the default. The system default key wrapping method can be specified using the DEFAULTWRAP parameter in the installation options data set. See the *z/OS Cryptographic Services ICSF System Programmer's Guide*.
WRAP-ENH	Use enhanced key wrapping method, which is compliant with the ANSI X9.24 standard.
WRAP-ECB	Use original key wrapping method, which uses ECB wrapping for DES key tokens.
Translation Control (optional, valid only with WRAP-ENH)	
ENH-ONLY	Restrict rewrapping of the *output_key_token*. Once the token has been wrapped with the enhanced method, it cannot be rewrapped using the original method.
Algorithm (One required, if the V0PYLD or V1PYLD keyword is specified)	
AES	Specifies that the input key is an AES key. Where used, the key-encrypting keys will be AES transport keys.
DES	Specifies that the input key is a DES key. Where used, the key-encrypting keys will be DES transport keys. This is the default.
HMAC	Specifies that the input key is an HMAC key. Where used, the key-encrypting keys will be AES transport keys.

input_key_length

Direction	Type
Input	Integer

The length of the *input_key_token* in bytes. The maximum value allowed is 900.

input_key_token

Direction	Type
Input/Output	String

A variable length string variable containing the key token to be translated or reformatted.

If the REFORMAT keyword is specified and the input_key_token is an AES CIPHER key (version X'05'), the key must have the following characteristics:

- Key-usage field 1 allows the key to be used for encryption and decryption and has no UDX bits set (UDX bits are not supported in version '04'X AES tokens)
- Key-usage field 2 allows the key to be used for Cipher Block Chaining (CBC) mode or Electronic Code Book (ECB) mode
- Key-management field 1 allows export using symmetric, unauthenticated asymmetric, and authenticated asymmetric transport keys, and allows export using DES, AES, and RSA transport keys
- Key-management field 2 indicates that the key is complete

If the REFORMAT and AES keywords are specified and *input_key_token* was encrypted under the old master key, the token will be returned encrypted under the current master key.

input_KEK_length

Direction	Type
Input	Integer

The length of the *input_KEK_identifier* in bytes. When the *input_KEK_identifier* is a token, the value must be between the actual length of the token and 725. When the *input_KEK_identifier* is a label, the value must be 64.

If the REFORMAT keyword is specified, and *input_key_token* is an AES key token, this parameter must be 0.

input_KEK_identifier

Direction	Type
Input/Output	String

A variable length string variable containing the internal key token or the key label of an internal key token record in the CKDS. The internal key token contains the key-encrypting key used to decipher the key.

If *input_KEK_length* is 0, this parameter is ignored.

If the TRANSLAT keyword is specified and the *input_key_token* is an external DES key, the *input_KEK_identifier* must be an internal DES token that contains a control vector that specifies an IMPORTER or IKEYXLAT key type. The control vector for an IMPORTER key must have the XLATE bit set to 1.

If the TRANSLAT, V0PYLD, or V1PYLD keyword is specified and the *input_key_token* is an external variable-length key token, the *input_KEK_identifier* must be an internal variable-length key token containing an IMPORTER key-encrypting key. The IMPORTER key must have the TRANSLAT bit on in key-usage field 1 of the token.

If the REFORMAT keyword is specified and *input_key_token* is an external DES key token, this parameter may be an IMPORTER, IKEYXLAT, EXPORTER, or OKEYXLAT key type.

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.

output_KEK_length

Direction	Type
Input	Integer

The length of the *output_KEK_identifier* in bytes. When the *output_KEK_identifier* is a token, the value must be between the actual length of the token and 725. When the *output_KEK_identifier* is a label, the value must be 64.

If the REFORMAT, V0PYLD, or V1PYLD keyword is specified, this value must be 0.

output_KEK_identifier

Direction	Type
Input/Output	String

A variable length string variable containing the internal key token or the key label of an internal key token record in the CKDS. The internal key token contains the key-encrypting key used to encipher the key.

If *output_KEK_length* is 0, this parameter is ignored.

If the *output_key_token* is an external DES key, the *output_KEK_identifier* must be an internal DES token that contains a control vector that specifies an EXPORTER or OKEYXLAT key type. The control vector for an EXPORTER key must have the XLATE bit set to 1.

If the *input_key_token* is an external variable-length key token, the *output_KEK_identifier* must be an internal variable-length key token containing an EXPORTER key-encrypting key. The EXPORTER key must have the TRANSLAT bit on in key-usage field 1 of the token.

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.

output_key_length

Direction	Type
Input/Output	Integer

On input, the length of the output area provided for the *output_key_token*. This must be between 64 and 900 bytes and provide sufficient space for the output key. On output, the parameter is updated with the length of the token copied to the *output_key_token*.

output_key_token

Direction	Type
Input/Output	String

If the REFORMAT keyword is specified and the *input_key_token* is an AES DATA key (version X'04'), *output_key_token* must contain an AES CIPHER key (version X'05') on input. This token must have the following characteristics:

- Algorithm is AES
- Key type CIPHER
- Key-usage field 2 either allows the key to be used for Cipher Block Chaining (CBC) mode or allows the key to be used for Electronic Code Book (ECB) mode

Otherwise, this field is ignored on input.

On output, a variable length string variable containing the key token that was translated or reformatted.

If the REFORMAT keyword is specified and the *input_key_token* is an AES DATA key (version X'04'), on output, *output_key_token* will be updated with the following characteristics:

- Key-usage field 1 allows the key to be used for encryption and decryption
- Key-management field 1 allows export using symmetric, unauthenticated asymmetric, and authenticated asymmetric transport keys, and allows export using DES, AES, and RSA transport keys
- Key-management field 2 indicates that the key is complete

Restrictions

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

Usage Notes

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

Access Control Points

This table lists the access control points in the domain role that control the function for this service.

Table 20. Key Translate2 Access Control Points

Access Control point	Function control
Key Translate2	Allows the Key Translate2 service to be functional.
Key Translate2 – Allow use of REFORMAT	Allows a key token to be rewrapped using one key-encrypting key.
Key Translate2 – Allow wrapping method override keywords	Allows the wrapping method keywords WRAP-ECB or WRAP-ENH to be used when the default key-wrapping method setting does not match the keyword.
Key Translate2 – Translate fixed to variable payload	Allows a key token with a fixed-length payload to be re-enciphered with a variable-length payload (V0PYLD).

When the **Key Translate2 - Disallow AES ver 5 to ver 4 conversion** access control point is enabled, a version 5 AES key token (variable-length token) can not be converted to a version 4 token.

If the output key-encrypting key identifier is a weaker key than the key being translated, 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 input key is a triple-length DATA key and the output key-encrypting key identifier key is a 16-byte key.

Required Hardware

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

Table 21. Key Translate2 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	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code. V0PYLD and V1PYLD keywords require the November 2013 or later licensed internal code. AES key support requires the September 2011 or later licensed internal code (LIC). Enhanced key token wrapping and HMAC key support requires the November 2010 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code. V0PYLD and V1PYLD keywords require the September 2013 or later licensed internal code.

Restrict Key Attribute (CSNBRKA and CSNERKA)

Use the Restrict Key Attribute callable service to modify an attribute of an internal or external CCA symmetric key-token.

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

Required Hardware

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

Table 22. Restrict Key Attribute required hardware

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

Table 22. Restrict Key Attribute required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC		This service is not supported.
IBM System z10 EC IBM System z10 BC		This service is not supported.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code. AES key support requires the September 2011 or later licensed internal code (LIC). Enhanced key token wrapping and HMAC key support requires the November 2010 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code.

Secure Key Import2 (CSNBSKI2 and CSNESKI2)

Use this service to encipher a variable-length symmetric key under the AES master key or an AES IMPORTER KEK, depending on the Key Form rule provided. This service returns variable-length CCA key tokens and uses the AESKW wrapping method.

Some key types are not directly supported by this service because there is no default key usage value. Also, some key usage flags are not supported by this service. These key types can be created by using the TOKEN keyword and a skeleton token from the Key Token Build2 service.

The following AES key types require TOKEN to be used: DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW.

The callable service can execute only when ICSF is in special secure mode, which is described in the 'Special Secure Mode' topic in *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

The callable service name for AMODE(64) is CSNESKI2.

Format

```
CALL CSNBSKI2(
            return_code,
            reason_code,
            exit_data_length,
            exit_data,
            rule_array_count,
            rule_array,
            clear_key_bit_length,
            clear_key,
            key_name_length,
            key_name,
            user_associated_data_length,
            user_associated_data,
```

```
key_encrypting_key_identifier_length,
key_encrypting_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. “ICSF and TSS Return and Reason Codes” on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes that indicate specific processing problems. “ICSF and TSS Return and Reason Codes” on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFF' (2 gigabytes). 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 3 or 4.

rule_array

Direction	Type
Input	String

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

Table 23. Keywords for Secure Key Import2 Control Information

Keyword	Meaning
Token algorithm (One Required)	
HMAC	The target key identifier is to be an HMAC key.
AES	The target key identifier is to be an AES key.
Key Form (One Required)	
OP	Specifies the key should be enciphered under the master key.
IM	Specifies the key should be enciphered under the key-encrypting key.
Key Type (One Required)	
CIPHER	The key type of the output token will be CIPHER. Only valid for AES algorithm.
EXPORTER	The key type of the output token will be EXPORTER. Only valid for AES algorithm.
IMPORTER	The key type of the output token will be IMPORTER. Only valid for AES algorithm.
MAC	MAC generation key.
MACVER	MAC verify key. Only valid for HMAC algorithm.
TOKEN	The key type will be determined from the key token supplied in the *target_key_identifier* parameter. ICSF does not check for the length of the key but uses the *clear_key_bit_length* parameter to determine the length of the key.
Payload version (One, optional) Note: This keyword overrides payload format version of any corresponding skeleton token.	
V0PYLD	The generated token will have the old variable length payload format. This is the default for AES CIPHER, EXPORTER, IMPORTER key types and is only valid with those key types.
V1PYLD	The generated token will have the new fixed length payload format. This is the default for AES MAC, PINPROT, PINCALC, PINPRW, and DKYGENKY key types. Not valid with the HMAC MAC key type.

clear_key_bit_length

Direction	Type
Input	Integer

The length of the value supplied in the *clear_key* parameter in bits. Valid lengths are 80 to 2048 for HMAC keys, and 128, 192, or 256 for AES keys.

clear_key

Direction	Type
Input	String

The value of the key to be imported. The value should be left justified and padded on the right with zeros to a byte boundary if the *clear_key_bit_length* is not a multiple of 8.

key_name_length

Direction	Type
Input	Integer

The length of the *key_name* parameter. Valid values are 0 and 64.

key_name

Direction	Type
Input	String

A 64-byte key store label to be stored in the associated data structure of the token.

user_associated_data_length

Direction	Type
Input	Integer

The length of the user-associated data. The valid values are 0 to 255 bytes.

user_associated_data

Direction	Type
Input	String

User-associated data to be stored in the associated data structure.

key_encrypting_key_identifier_length

Direction	Type
Input	Integer

The byte length of the *key_encrypting_key_identifier* parameter. When Key Form is OP, the value must be 0. When Key Form is IM, the value must be between the actual length of the token and 725 when *key_encrypting_key_identifier* is a token. The value must be 64 when *key_encrypting_key_identifier* is a label.

key_encrypting_key_identifier

Direction	Type
Input/Output	String

When the Key Form rule is OP, *key_encrypting_key_identifier* is ignored. When the Key Form rule is IM, *key_encrypting_key_identifier* contains an internal key token containing the AES importer key-encrypting key or a key label.

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

target_key_identifier_length

Direction	Type
Input/Output	Integer

On input, the byte length of the buffer for the *target_key_identifier* parameter. The buffer must be large enough to receive the target key token. The maximum value is 900 bytes.

On output, the parameter will hold the actual length of the target key token.

target_key_identifier

Direction	Type
Input/Output	String

The output key token. On input, this parameter is ignored except when the Key Type keyword is TOKEN. If you specify the TOKEN keyword, then this field contains a valid token of the key type you want to import. On output, when Key Form is OP, this will be an internal variable-length symmetric token. When Key Form is IM, this will be an external variable-length symmetric token. See *rule_array* for a list of valid key types.

Access Control Points

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

Table 24. Required access control points for Secure Key Import2

Key Form	Access control point
OP	Secure Key Import2 – OP
IM	Secure Key Import2 – IM

When the **Symmetric Key Import2 - disallow weak import** access control point is enabled, a key token wrapped with a weaker key will not be imported. When the **Warn when weak wrap - Transport keys** access control point is enabled, the reason code will indicate when the wrapping key is weaker than the key being imported.

Required Hardware

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

Table 25. Secure Key Import2 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.

Table 25. Secure Key Import2 required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code. V0PYLD and V1PYLD keywords require the November 2013 or later licensed internal code. 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).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code. V0PYLD and V1PYLD keywords require the September 2013 or later licensed internal code (LIC).

Symmetric Key Export (CSNDSYX and CSNFSYX)

Use the symmetric key export callable service to transfer an application-supplied AES, DES or variable-length symmetric key token key from encryption under a master key to encryption under an application-supplied RSA public key or AES EXPORTER key. The application-supplied key must be an ICSF AES, DES, or HMAC internal key token or the label of such a token in the CKDS. The Symmetric Key Import or Symmetric Key Import2 callable services can import the key encrypted under the RSA public key or AES EXPORTER at the receiving node.

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

Required Hardware

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

Table 26. Symmetric key export 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. Encrypted AES keys are not supported. The AESKW, AESKWCV, HMAC, and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP.

Table 26. Symmetric key export required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the November 2007 or later licensed internal code (LIC). Encrypted AES key support requires the November 2008 or later licensed internal code (LIC). The AESKW, AESKWCV, HMAC, and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP.
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the November 2007 or later licensed internal code (LIC). Encrypted AES key support requires the November 2008 or later licensed internal code (LIC). The AESKW, AESKWCV, HMAC, and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP.
	Crypto Express3 Coprocessor	The AESKW, AESKWCV, HMAC, and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP.
IBM zEnterprise 196 IBM zEnterprise 114	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code. Variable-length AES Keys, the AESKW method, and PKCSOAEP with the SHA-256 hash method require the September 2011 or later licensed internal code (LIC). HMAC key support requires the November 2010 or later licensed internal code (LIC). The AESKWCV keyword is not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code. AESKWCV requires the September 2013 or later licensed internal code (LIC).

Symmetric Key Import2 (CSNDSYI2 and CSNFSYI2)

Use the Symmetric Key Import2 callable service to import an HMAC, AES or DES key enciphered under an RSA public key or AES EXPORTER key. It returns the key in operational form, enciphered under the master key.

This service returns a variable-length CCA key token wrapped using the mode configured as the default wrapping mode, either enhanced wrapping mode (WRAP-ENH) or original ECB wrapping mode (WRAP-ECB).

The callable service name for AMODE(64) is CSNFSYI2.

Required Hardware

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

Table 27. Symmetric key import2 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	Crypto Express3 Coprocessor	HMAC key support requires the November 2010 or later licensed internal code (LIC). AES key support and the AESKW wrapping method require the September 2011 or later licensed internal code (LIC). DK AES PIN key support requires the November 2013 or later licensed internal code. DES, AESKWCV, USECONFG, WRAP-ECB, WRAP-ENH and ENH-ONLY keywords are not supported.
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code. DES, AESKWCV, USECONFG, WRAP-ECB, WRAP-ENH and ENH-ONLY keywords require the September 2013 or later licensed internal code (LIC).

Financial Services for DK PIN Methods

This topic provides the new callable services that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods:

- "DK PAN Modify in Transaction (CSNBDPMT and CSNEDPMT)" on page 68
- "DK PIN Change (CSNBDPC and CSNEDPC)" on page 75
- "DK PIN Verify (CSNBDPV and CSNEDPV)" on page 87
- "DK Random PIN Generate (CSNBDRPG and CSNEDRPG)" on page 91

Weak PIN table

The DK PIN methods support the use a table of weak PINs. Services that generate PINs compare the generated PIN against the table and if the PIN is in the table, the service generates a different PIN. Services that change PINs compare the new PIN against the table and if the new PIN is in the table, the service fails.

Weak PIN tables can be stored in the cryptographic coprocessors for use by callable services. Only tables that have been activated can be used. A TKE Workstation is required to manage the tables in the coprocessors.

Note: ICSF routes work to all active coprocessors based on work load. All coprocessors must have the same set of PINs.

DK PIN methods

The DK PIN methods use a PIN Reference Value (PRW) to verify PINs rather than regenerating the PIN from customer account data. The PRW is generated by concatenating the customer PAN data, the issuer card data, the PIN length, the PIN, and a 4-byte random number and encrypting using a PRW key with the GENONLY key usage. The PRW and random number are the output of the generation. The PIN is verified by generating the PRW using a PRW key with the VERIFY key usage and comparing it against the supplied PRW and random number.

DK PAN Modify in Transaction (CSNBDPMT and CSNEDPMT)

Use the DK PAN Modify in Transaction callable service to generate a new PIN reference value (PRW) for an existing PIN when a merger has occurred and the account information has changed. The inputs include the current PIN, the account information (PAN and card data) for the current, and the new account.

The DK PRW CMAC Generate service is called prior to this service to generate the MAC of the changed account information. If the MAC associated with the account information does not verify, the service fails.

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

Format

```
CALL CSNBPDPMT(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             current_PAN_data_length,
             current_PAN_data,
             new_PAN_data_length,
             new_PAN_data,
             current_card_p_data_length,
             current_card_p_data,
             current_card_t_data_length,
             current_card_t_data,
             new_card_p_data_length,
             new_card_p_data,
             new_card_t_data_length,
             new_card_t_data,
             CMAC_FUS_length,
             CMAC_FUS,
             ISO_encrypted_PIN_block_length,
             ISO_encrypted_PIN_block,
             current_PIN_reference_value_length,
             current_PIN_reference_value,
             current_PRW_random_number_length,
             current_PRW_random_number,
             CMAC_FUS_key_identifier_length,
             CMAC_FUS_key_identifier,
             IPIN_encryption_key_identifier_length,
             IPIN_encryption_key_identifier,
             PRW_key_identifier_length,
             PRW_key_identifier,
             new_PRW_key_identifier_length,
             new_PRW_key_identifier,
```

```
new_PIN_reference_value_length,
new_PIN_reference_value,
new_PRW_random_number_length,
new_PRW_random_number)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. “ICSF and TSS Return and Reason Codes” on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. “ICSF and TSS Return and Reason Codes” on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFF' (2 gigabytes). The data is identified in the *exit_data* parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	Character

Keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks. There are no keywords for this service.

current_PAN_data_length

Direction	Type
Input	Integer

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

current_PAN_data

Direction	Type
Input	Character

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

new_PAN_data_length

Direction	Type
Input	Integer

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

new_PAN_data

Direction	Type
Input	Character

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

current_card_p_data_length

Direction	Type
Input	Integer

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

current_card_p_data

Direction	Type
Input	String

The time-invariant card data (CDp) of the current account, determined by the card issuer.

current_card_t_data_length

Direction	Type
Input	Integer

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

current_card_t_data

Direction	Type
Input	String

The time-invariant card data (CDp) of the current account, determined by the card issuer.

new_card_p_data_length

Direction	Type
Input	Integer

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

new_card_p_data

Direction	Type
Input	String

The time-invariant card data (CDp) of the current account, determined by the card issuer.

new_card_t_data_length

Direction	Type
Input	Integer

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

new_card_t_data

Direction	Type
Input	String

The time-invariant card data (CDp) of the current account, determined by the card issuer.

CMAC_FUS_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *CMAC_FUS* parameter. The value must be between 8 and 16, inclusive.

CMAC_FUS

Direction	Type
Input	String

The 8-byte to 16-byte MAC that was of the current and new PANs and card data strings and PIN reference values. The MAC is generated using the DK PRW CMAC Generate service.

ISO_encrypted_PIN_block_length

Direction	Type
Input	Integer

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

ISO_encrypted_PIN_block

Direction	Type
Input	String

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

current_PIN_reference_value_length

Direction	Type
Input	Integer

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

current_PIN_reference_value

Direction	Type
Input	String

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

current_PRW_random_number_length

Direction	Type
Input	Integer

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

current_PRW_random_number

Direction	Type
Input	String

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

CMAC_FUS_key_identifier_length

Direction	Type
Input	Integer

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

CMAC_FUS_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to verify the *CMAC_FUS* value. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be AES, the key type must be MAC, and the key usage fields must indicate VERIFY, CMAC, and DKPINAD2.

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

IPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

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

IPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the *encrypted_PIN_block*. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be DES and the key type must be IPINENC.

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

PRW_key_identifier_length

Direction	Type
Input	Integer

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

PRW_key_identifier

Direction	Type
Input/Output	String

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

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

new_PRW_key_identifier_length

Direction	Type
Input	Integer

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

new_PRW_key_identifier

Direction	Type
Input/Output	String

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

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

new_PIN_reference_value_length

Direction	Type
Input/Output	Integer

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

new_PIN_reference_value

Direction	Type
Output	String

The 16-byte new PIN reference value.

new_PRW_random_number_length

Direction	Type
Input/Output	Integer

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

new_PRW_random_number

Direction	Type
Output	String

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

Usage Notes

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

Access Control Points

The **DK PAN Modify in Transaction** access control point in the domain role controls the function of this service.

Required Hardware

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

Table 28. DK PAN Modify in Transaction 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	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code (LIC).

DK PIN Change (CSNBDPC and CSNEDPC)

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

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

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

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

Format

```
CALL CSNBDPC(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
```

```
rule_array,
PAN_data_length,
PAN_data,
card_p_data_length,
card_p_data,
card_t_data_length,
card_t_data,
cur_ISO1_PIN_block_length,
cur_ISO1_PIN_block,
new_ISO1_PIN_block_length,
new_ISO1_PIN_block,
card_script_data_length,
card_script_data,
script_offset,
script_offset_field_length,
script_initialization_vector_length,
script_initialization_vector,
output_PIN_profile,
PIN_reference_value_length,
PIN_reference_value,
PRW_random_number_length,
PRW_random_number,
PRW_key_identifier_length,
PRW_key_identifier,
cur_IPIN_encryption_key_identifier_length,
cur_IPIN_encryption_key_identifier,
new_IPIN_encryption_key_identifier_length,
new_IPIN_encryption_key_identifier,
script_key_identifier_length,
script_key_identifier,
script_MAC_key_identifier_length,
script_MAC_key_identifier,
new_PRW_key_identifier_length,
new_PRW_key_identifier,
OPIN_encryption_key_identifier_length,
OPIN_encryption_key_identifier,
OEPB_MAC_key_identifier_length,
OEPB_MAC_key_identifier,
script_length,
script,
script_MAC_length,
script_MAC,
new_PIN_reference_value_length,
new_PIN_reference_value,
new_PRW_random_number_length,
new_PRW_random_number,
output_encrypted_PIN_block_length,
output_encrypted_PIN_block,
PIN_block_MAC_length,
PIN_block_MAC)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. “ICSF and TSS Return and Reason Codes” on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. "ICSF and TSS Return and Reason Codes" on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFF' (2 gigabytes). The data is identified in the *exit_data* parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	Character

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

Table 29. Rule array keywords for the Card Replace PRW Generate Service

Keyword	Meaning
PIN Block output selection keyword (One, optional)	
NOEPB	Do not return an encrypted PIN block (EPB). This is the default value.
EPB	Return an encrypted PIN block and a MAC to verify the encrypted PIN block.
Script selection keyword (One, optional)	
NOSCRIPT	Do not return an encrypted SMPIN message with a MAC. This is the default value.
TDES-CBC	Use CBC mode to encrypt the script.
TDES-ECB	Use ECB mode to encrypt the script.

Table 29. Rule array keywords for the Card Replace PRW Generate Service (continued)

Keyword	Meaning
Pin encryption keyword (One, optional) Only valid if TDES-CBC or TDES-ECB is selected above.	
CLEARPIN	Do not encrypt the PIN prior to inserting in the script block. This is the default value.
SELF-ENC	Copy the PIN-block self-encrypted to the clear PIN block within the clear output message. Use this rule array keyword to specify that the 8-byte PIN block shall be used as a DES key to encrypt the PIN block. The service copies the self-encrypted PIN block to the clear PIN block in the output message.
MAC Ciphering Method (One, optional) Only valid if TDES-CBC or TDES-ECB is selected above.	
EMVMACD	Specifies the EMV-related message-padding and calculation method.
TDES-MAC	Specifies the ANS X9.9 Option 1 (binary data) procedure and a CBC Triple-DES encryption of the data.
X9.19OPT	Specifies the ANS X9.19 Optional Procedure. A double-length key is required. This is the default value.
MAC Length and presentation (One, optional) Only valid if TDES-CBC or TDES-ECB is selected above.	
MACLEN8	Specifies a 8-byte MAC. This is the default value.

PAN_data_length

Direction	Type
Input	Integer

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

PAN_data

Direction	Type
Input	Character

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

card_p_data_length

Direction	Type
Input	Integer

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

card_p_data

Direction	Type
Input	String

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

card_t_data_length

Direction	Type
Input	Integer

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

card_t_data

Direction	Type
Input	String

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

cur_ISO1_PIN_block_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *cur_ISO1_PIN_block* parameter. This value must be 8.

cur_ISO1_PIN_block

Direction	Type
Input	String

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

new_ISO1_PIN_block_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *new_ISO1_PIN_block* parameter. This value must be 8.

new_ISO1_PIN_block

Direction	Type
Input	String

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

card_script_data_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *card_script_data* parameter. If NOSCRIPT is specified in the rule array, this value must be 0. Otherwise, the value must be no greater than 4096 and a multiple of 8.

card_script_data

Direction	Type
Input	String

The clear text string to be updated with the clear PIN block and encrypted.

script_offset

Direction	Type
Input	Integer

The offset to the location for the PIN block in the script. Specify the first byte of the clear text as offset 0. This offset plus the value of *script_offset_field_length* must be less than or equal to the *card_script_data_length*. If NOSCRIPT is specified in the rule array, this parameter is ignored.

script_offset_field_length

Direction	Type
Input	Integer

The length of the field within *card_script_text* parameter at *script_offset* where the new PIN value is to be placed. Length must be 8. The PIN block must fit entirely within the *card_script_text*. If NOSCRIPT is specified in the rule array, this parameter is ignored.

script_initialization_vector_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *script_initialization_vector* parameter. If NOSCRIPT or TDES-ECB is specified in the rule array, this value must be 0. Otherwise, it must be 8.

script_initialization_vector

Direction	Type
Input	String

The 8-byte initialization data for encrypting the script. If the *script_initialization_vector_length* is 0, this parameter is ignored.

output_PIN_profile

Direction	Type
Input	String

A 24-byte string containing the PIN profile, including the PIN block format for the script. See 'The PIN Profile' for additional information. You can use

PIN-block formats ISO-0, ISO-1, ISO-2, and ISO-3 with this service. If NOSCRIPT is specified in the rule array, this parameter is ignored.

PIN_reference_value_length

Direction	Type
Input	Integer

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

PIN_reference_value

Direction	Type
Input	String

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

PRW_random_number_length

Direction	Type
Input	Integer

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

PRW_random_number

Direction	Type
Input	String

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

PRW_key_identifier_length

Direction	Type
Input	Integer

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

PRW_key_identifier

Direction	Type
Input/Output	String

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

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

cur_IPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

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

cur_IPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the PIN_block containing the current PIN. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be DES and the key type must be IPINENC.

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

new_IPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

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

new_IPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the PIN_block containing the new PIN. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be DES and the key type must be IPINENC.

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

script_key_identifier_length

Direction	Type
Input	Integer

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

script_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the script. 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 DES, the key type must be SECMSG key type with the SMPIN usage bit (CV bit 19) set to B'1'.

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

script_MAC_key_identifier_length

Direction	Type
Input	Integer

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

script_MAC_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to generate the MAC of the script. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be DES, the key type must be MAC, and the key must be double-length.

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

new_PRW_key_identifier_length

Direction	Type
Input	Integer

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

new_PRW_key_identifier

Direction	Type
Input/Output	String

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

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

OPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

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

OPIN_encryption_key_identifier

Direction	Type
Input/Output	String

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

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

OEPB_MAC_key_identifier_length

Direction	Type
Input	Integer

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

OEPB_MAC_key_identifier

Direction	Type
Input/Output	String

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

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

script_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *script* parameter. If the rule array specifies TDES-CBC or TDES-ECB, this value must be at least as long as the *script* parameter. Otherwise, it must be 0.

script

Direction	Type
Output	String

The encrypted output script. The length of the field must be at least as long as the input script.

script_MAC_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *script_MAC* parameter. If the NOSCRIPT keyword is selected, this value must be 0. Otherwise, this must be at least 8.

script_MAC

Direction	Type
Output	String

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

new_PIN_reference_value_length

Direction	Type
Input/Output	Integer

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

new_PIN_reference_value

Direction	Type
Output	String

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

new_PRW_random_number_length

Direction	Type
Input/Output	Integer

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

new_PRW_random_number

Direction	Type
Output	String

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

output_encrypted_PIN_block_length

Direction	Type
Input/Output	Integer

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

output_encrypted_PIN_block

Direction	Type
Output	String

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

PIN_block_MAC_length

Direction	Type
Input/Output	Integer

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

PIN_block_MAC

Direction	Type
Output	String

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

Usage Notes

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

Access Control Points

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

Required Hardware

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

Table 30. DK PIN Change 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 30. DK PIN Change 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 key support requires the November 2013 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code (LIC).

DK PIN Verify (CSNBDPV and CSNEDPV)

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

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

Format

```
CALL CSNBDPV(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             PAN_data_length,
             PAN_data,
             card_data_length,
             card_data,
             PIN_reference_value_length,
             PIN_reference_value,
             PRW_random_number_length,
             PRW_random_number,
             ISO_encrypted_PIN_block_length,
             ISO_encrypted_PIN_block,
             PRW_key_identifier_length,
             PRW_key_identifier,
             IPIN_encryption_key_identifier_length,
             IPIN_encryption_key_identifier)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. “ICSF and TSS Return and Reason Codes” on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. “ICSF and TSS Return and Reason Codes” on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFF' (2 gigabytes). The data is identified in the *exit_data* parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	Character

Keywords that provide control information to the callable service. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks. There are no keywords for this service.

PAN_data_length

Direction	Type
Input	Integer

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

PAN_data

Direction	Type
Input	Character

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

card_data_length

Direction	Type
Input	Integer

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

card_data

Direction	Type
Input	String

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

PIN_reference_value_length

Direction	Type
Input	Integer

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

PIN_reference_value

Direction	Type
Input	String

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

PRW_random_number_length

Direction	Type
Input	Integer

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

PRW_random_number

Direction	Type
Input	String

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

ISO_encrypted_PIN_block_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *ISO_encrypted_PIN_block* parameter. This value must be 8.

ISO_encrypted_PIN_block

Direction	Type
Input	String

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

PRW_key_identifier_length

Direction	Type
Input	Integer

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

PRW_key_identifier

Direction	Type
Input/Output	String

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

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

IPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

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

IPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to decrypt the PIN_block. The key identifier is an operational token or the key label of an operational token in key storage. The key algorithm of this key must be DES and the key type must be IPINENC.

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

Usage Notes

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

Access Control Points

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

Required Hardware

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

Table 31. DK PIN Verify 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	Crypto Express3 Coprocessor	DK AES PIN key support requires the November 2013 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code (LIC).

DK Random PIN Generate (CSNBDRPG and CSNEDRPG)

Use the DK Random PIN Generate callable service to generate a PIN and a PIN reference value using the random process. After the PIN is generated, a PIN reference value (PRW) is created. The PIN reference value is used to verify the PIN in other processes.

Note: Regarding weak PINs, if the PIN which is generated appears in the weak PIN table, the generation process is modified and re-tried until a valid PIN is generated.

You can use this service to perform the following tasks:

- Generate an encrypted PIN block in PBF-1 format with a PIN print key to be printed on a PIN mailer.
- Generate a PIN reference value which can be used to verify the PIN.
- Optionally, generate an encrypted PIN block in PBF-1 format to be stored for later use in personalizing replacement cards, along with a verifying CMAC over the encrypted block and additional card data.

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

Format

```
CALL CSNBDRPG(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             PAN_data_length,
             PAN_data,
             card_p_data_length,
             card_p_data,
             card_t_data_length,
             card_t_data,
```

```
PIN_length,
PRW_key_identifier_length,
PRW_key_identifier,
PIN_print_key_identifier_length,
PIN_print_key_identifier,
OPIN_encryption_key_identifier_length,
OPIN_encryption_key_identifier,
OEPB_MAC_key_identifier_length,
OEPB_MAC_key_identifier,
PIN_reference_value_length,
PIN_reference_value,
PRW_random_number_length,
PRW_random_number,
PIN_print_block_length,
PIN_print_block,
encrypted_PIN_block_length,
encrypted_PIN_block,
PIN_block_MAC_length,
PIN_block_MAC)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. "ICSF and TSS Return and Reason Codes" on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. "ICSF and TSS Return and Reason Codes" on page 120 lists the reason codes.

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The length can be from X'00000000' to X'7FFFFFFF' (2 gigabytes). The data is identified in the *exit_data* parameter.

exit_data

Direction	Type
Input/Output	String

The data that is passed to the installation exit.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	Character

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

Table 32. Rule array keywords for the PIN Generate2 with Reference Value Service

Keyword	Meaning
PIN Block output selection keyword (One, optional)	
NOEPB	Do not return an encrypted PIN block (EPB). This is the default value.
EPB	Return an encrypted PIN block.

PAN_data_length

Direction	Type
Input	Integer

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

PAN_data

Direction	Type
Input	Character

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

card_p_data_length

Direction	Type
Input	Integer

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

card_p_data

Direction	Type
Input	String

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

card_t_data_length

Direction	Type
Input	Integer

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

card_t_data

Direction	Type
Input	String

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

PIN_length

Direction	Type
Input	Integer

Specifies the length of the PIN to be generated. This value must be between 4 and 12, inclusive.

PRW_key_identifier_length

Direction	Type
Input	Integer

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

PRW_key_identifier

Direction	Type
Input/Output	String

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

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

PIN_print_key_identifier_length

Direction	Type
Input	Integer

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

PIN_print_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to wrap the PIN for printing. 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 PINPROT, and the key usage fields must indicate ENCRYPT, CBC, and DKPINOPP.

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

OPIN_encryption_key_identifier_length

Direction	Type
Input	Integer

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

OPIN_encryption_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to wrap the PIN block. The key identifier is an operational token or the key label of an operational token in key storage. If the rule array indicates that no encrypted PIN block is to be returned, this parameter is ignored. The key algorithm of this key must be AES, the key type must be PINPROT, and the key usage fields must indicate ENCRYPT, CBC, and DKPINOP.

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

OEPB_MAC_key_identifier_length

Direction	Type
Input	Integer

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

OEPB_MAC_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to generate the MAC of the PIN block. The key identifier is an operational token or the key label of an operational token in key storage. If the rule array indicates that no encrypted PIN block is to be

returned, this parameter is ignored. The key algorithm of this key must be AES, the key type must be MAC, the key usage fields must indicate GENONLY, CMAC, and DKPINOP.

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

PIN_reference_value_length

Direction	Type
Input/Output	Integer

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

PIN_reference_value

Direction	Type
Output	String

The 16-byte calculated PIN reference value.

PRW_random_number_length

Direction	Type
Input/Output	Integer

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

PRW_random_number

Direction	Type
Output	String

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

PIN_print_block_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *PIN_print_block* parameter. It must be at least 32. On output, it will be set to 32.

PIN_print_block

Direction	Type
Output	String

The 32-byte encrypted PIN block to be passed to the PIN mailer function.

encrypted_PIN_block_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *encrypted_PIN_block* parameter. If the rule array indicates that no encrypted PIN block should be returned, this value must be 0. Otherwise, it should be at least 32.

encrypted_PIN_block

Direction	Type
Output	String

The 32-byte encrypted PIN block PBF-1 format. This parameter is ignored if no encrypted PIN block is returned.

PIN_block_MAC_length

Direction	Type
Input/Output	Integer

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

PIN_block_MAC

Direction	Type
Output	String

The 8-byte CMAC of the encrypted PIN block. This parameter is ignored if no encrypted PIN block is returned.

Usage Notes

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

Access Control Points

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

Required Hardware

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

Table 33. DK Random PIN Generate 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	Crypto Express3 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code (LIC).

Table 33. DK Random PIN Generate required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	DK AES PIN key support requires the September 2013 or later licensed internal code (LIC).

Utilities

This topic describes the updated callable service that supports the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods:

- "ICSF Query Facility (CSFIQF and CSFIQF6)"

ICSF Query Facility (CSFIQF and CSFIQF6)

Use this utility to retrieve information about ICSF, the cryptographic coprocessors and the CCA code in the coprocessors. This information includes:

- General information about ICSF
- General information about CCA code in a coprocessor
- Export control information from a coprocessor
- Diagnostic information from a coprocessor

Coprocessor information requests may be directed to a specific ONLINE or ACTIVE coprocessor or any ACTIVE coprocessor.

This service has an interface similar to the IBM 4765 service CSUACFQ. Instead of the output being returned in the rule array, there is a separate output area. The format of the data returned remains the same. This service supports a subset of the keywords supported by CSUACFQ. For the same supported keywords, CSFIQF and CSUACFQ return the same coprocessor-specific information. The service returns information elements in the *returned_data* field and updates the *returned_data_length* with the actual length of the output *returned_data* field.

The callable service name for AMODE(64) invocation is CSFIQF6.

Format

```
CALL CSFIQF(
            return_code,
            reason_code,
            exit_data_length,
            exit_data,
            rule_array_count,
            rule_array,
            returned_data_length,
            returned_data,
            reserved_data_length,
            reserved_data)
```

Parameters

return_code

Direction	Type
Output	Integer

The return code specifies the general result of the callable service. "ICSF and TSS Return and Reason Codes" on page 120 lists the return codes.

reason_code

Direction	Type
Output	Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. "ICSF and TSS Return and Reason Codes" on page 120 lists the reason codes.

exit_data_length

Direction	Type
Ignored	Integer

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

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

The number of keywords you are supplying in *rule_array*. Value must be 1, 2 or 3.

rule_array

Direction	Type
Input	String

Keywords that provide control information to callable services. The keywords are left-justified in an 8-byte field and padded on the right with blanks. The keywords must be in contiguous storage. Specify one or two of the values in Table 34.

Table 34. Keywords for ICSF Query Service

Keyword	Meaning
Coprocessor (optional) - parameter is ignored for ICSFSTAT, ICSFST2, and ICSFSP11	
COPROCxx	Specifies the specific coprocessor to execute the request. xx may be 00 through 63 inclusive. This may be the processor number of any coprocessor. The processor number of any accelerator is not supported. If specified with rule STATP11, the processor number must be that of a Enterprise PKCS #11 coprocessor. For all other rules, it must be that of a CCA coprocessor.

Table 34. Keywords for ICSF Query Service (continued)

Keyword	Meaning
ANY	Process request on any ACTIVE cryptographic coprocessor. This is the default.
nnnnnnnn	Specifies the 8-byte serial number of the coprocessor to execute the request. If specified with rule STATP11, the processor number must be that of a Enterprise PKCS #11 coprocessor. For all other rules, it must be that of a CCA coprocessor.
Information to return (required)	
ICSFSTAT	Get ICSF related status information.
ICSFST2	Get coprocessor-related basic status information.
ICSFSP11	Get ICSF-related PKCS #11 status information
NUM-DECT	Get the number of bytes of storage required for the output of a STATDECT request.
STATAES	Get status information on AES enablement and the AES master key registers.
STATCCA	Get CCA-related status information.
STATCCAE	Get CCA-related extended status information.
STATCARD	Get coprocessor-related basic status information.
STATDECT	Get the PIN decimalization tables loaded. The format of the data is shown under the *returned_data* parameter. The length of the data is 20 bytes per decimalization table. The NUM-DECT option will return the storage required for this option. The maximum length of the data is 2000 bytes.
STATDIAG	Get coprocessor-related basic status information.
STATAPKA	Get status information on ECC enablement and the ECC master key registers.
STATEID	Get coprocessor-related basic status information.
STATEXPT	Get coprocessor-related basic status information.
WRAPMTHD	Get coprocessor-related default configuration setting for the wrapping method.
STATP11	Get Enterprise PKCS #11 coprocessor-related status information.
STATWPIN	Get the weak PIN table loaded. The format of the data is shown under the *returned_data* parameter. The table is up to 460 bytes long.
SIZEWPIN	Get the number of bytes of storage required for the output of a STATWPIN request.
Additional Master Key Information (optional) - rule is only allowed with STATCCA or STATCCAE	
MOREMKS	Return additional master key information

returned_data_length

Direction	Type
Input/Output	Integer

The length of the *returned_data* parameter. Currently, the value must be at least eight times the number of elements returned for the *rule_array* keyword specified. Allow additional space for future enhancements. On output, this field will contain the actual length of the data returned.

returned_data

Direction	Type
Output	String/Integer

This field will contain the output from the service. The format of the output depends on the *rule_array* keyword. The format of the data is defined in the tables below, which describe the output for each keyword.

When the format is 8-byte elements that contain numbers, those numbers are represented by numeric characters which are left-justified and padded on the right with space characters. For example, a *returned_data* element which contains the number two will contain the character string '2'.

For option NUM-DECT, the output is a 4-byte integer.

The output *returned_data* for the ICSFSTAT keyword is defined in Table 35.

Table 35. Output for option ICSFSTAT

Element Number	Name	Description
1	FMID	8-byte ICSF FMID
2	ICSF Status Field 1	Status of ICSF **Number** **Meaning** **0** ICSF started **1** ICSF initialized (CCVINIT is on) **2** SYM-MK (DES master key) valid (CCVTMK is on) **3** PKA callable services enabled (see "Usage Notes" on page 119)
3	ICSF Status Field 2	Status of ICSF **Number** **Meaning** **0** 64-bit callers not supported **1** 64-bit callers supported **2** 64-bit callers supported, and a TKDS has been specified for the storage of persistent PKCS #11 objects.

Table 35. Output for option ICSFSTAT (continued)

Element Number	Name	Description
4	CPACF	CPACF availability **Number** **Meaning** **0** CPACF not available **1** SHA-1 available only **2** DES/TDES enabled **3** SHA-224 and SHA-256 are available **4** SHA-224 and SHA-256, DES and TDES are available **5** SHA-384 and SHA-512 are available **6** SHA-384 and SHA-512, DES and TDES are available **7** Encrypted CPACF functions available. **8** OFB, CFB, and GCM CPACF functions are available.
5	AES	AES availability for clear keys **Number** **Meaning** **0** AES not available **1** AES software only **2** AES-128 **3** AES-192 and AES-256
6	DSA	DSA algorithm availability **Number** **Meaning** **0** DSA not available **1** DSA 1024 key size **2** DSA 2048 key size
7	RSA Signature	RSA Signature key length **Number** **Meaning** **0** RSA not available **1** RSA 1024 key size **2** RSA 2048 key size **3** RSA 4096 key size

Table 35. Output for option ICSFSTAT (continued)

Element Number	Name	Description
8	RSA Key Management	RSA Key Management key length **Number** **Meaning** **0** RSA not available **1** RSA 1024 key size **2** RSA 2048 key size **3** RSA 4096 key size
9	RSA Key Generate	RSA Key Generate **Number** **Meaning** **0** Service not available **1** Service available - 2048 bit modulus **2** Service available - 4096 bit modulus
10	Accelerators	Availability of clear RSA key accelerators **Number** **Meaning** **0** Not available **1** At least one available for application use.
11	Accelerator Key Size	Clear key size supported by Accelerators. There must be at least one Accelerator available for use for this field to contain valid information. **Number** **Meaning** **0** RSA-ME key size of 2048, CRT key size of 2048. **1** RSA-ME key size of 4096, CRT key size of 4096.
12	ICSF Status Field 3	An 8-byte numeric character string. The first character in this string indicates the current Special Secure Mode (SSM) setting. **Number** **Meaning** **0** SSM not allowed. **1** SSM allowed.

The output *returned_data* for the ICSFSP11 keyword is defined in Table 36 on page 104.

Table 36. Output for option ICSFSP11

Element Number	Name	Description
1	P11-MK State	Status of the P11-MK **Number** **Meaning** **0** P11-MK not active **1** P11-MK active
2	FIPS Mode	ICSF PKCS #11 FIPS mode **Number** **Meaning** **0** FIPS no enforcement mode **1** FIPS compatibility mode **2** FIPS mode
3-12	Future use	Currently blanks

The output *returned_data* for the ICSFST2 keyword is defined in Table 37.

Table 37. Output for option ICSFST2

Element Number	Name	Description
1	Version	Version of the ICSFST2 returned_data. Initial value is 1. It covers elements 1 through 12.
2	FMID	8–byte ICSF FMID.
3	ICSF Status Field 1	Status of ICSF **Number** **Meaning** **0** PKA callable services disabled **1** PKA callable services enabled (see "Usage Notes" on page 119)
4	ICSF Status Field 2	Status of ICSF **Number** **Meaning** **0** PKCS #11 is not available **1** PKCS #11 is available
5	ICSF Status Field 3	Status of ICSF **Number** **Meaning** **0** ICSF started **1** ICSF initialized **2** AES master key valid

Table 37. Output for option ICSFST2 (continued)

Element Number	Name	Description
6	ICSF Status Field 4	Status of ICSF **Number** **Meaning** **0** Secure key AES not available **1** Secure key AES is available
7	ICSF Status Field 5	An 8-character numeric character string summarizing the current Key Store Policy. The first character in this string indicates if Key Token Authorization Checking controls have been enabled for the CKDS in either warning or fail mode, and, if so, if the Default Key Label Checking control has also been enabled. The numbers that can appear in the first character of this string are: **Number** **Meaning** **0** Key Token Authorization Checking is not enabled for the CKDS. **1** Key Token Authorization Checking for CKDS is enabled in FAIL mode. **Key Store Policy is active for CKDS.** Default Key Label Checking is not enabled. **2** Key Token Authorization Checking for CKDS is enabled in WARN mode. **Key Store Policy is active for CKDS.** Default Key Label Checking is not enabled. **3** Key Token Authorization Checking for CKDS is enabled in FAIL mode. **Key Store Policy is active for CKDS.** Default Key Label Checking is also enabled. **4** Key Token Authorization Checking for CKDS is enabled in WARN mode. **Key Store Policy is active for CKDS.** Default Key Label Checking is also enabled.

Table 37. Output for option ICSFST2 (continued)

Element Number	Name	Description
		The second character in this string indicates if Duplicate Key Token Checking controls have been enabled for the CKDS. The numbers that can appear in the second character of this string are: **Number** **Meaning** **0** Duplicate Key Token Checking is not enabled for the CKDS. **1** Duplicate Key Token Checking is enabled for the CKDS. **Key Store Policy is active for CKDS.** The third character in this string indicates if Key Token Authorization Checking controls have been enabled for the PKDS in either warning or fail mode, and, if so, if the Default Key Label Checking control has also been enabled. The numbers that can appear in the third character of this string are: **Number** **Meaning** **0** Key Token Authorization Checking is not enabled for the PKDS. **1** Key Token Authorization Checking for PKDS is enabled in FAIL mode. **Key Store Policy is active for PKDS.** Default Key Label Checking is not enabled. **2** Key Token Authorization Checking for PKDS is enabled in WARN mode. **Key Store Policy is active for PKDS.** Default Key Label Checking is not enabled. **3** Key Token Authorization Checking for PKDS is enabled in FAIL mode. **Key Store Policy is active for PKDS.** Default Key Label Checking is also enabled. **4** Key Token Authorization Checking for PKDS is enabled in WARN mode. **Key Store Policy is active for PKDS.** Default Key Label Checking is also enabled.

Table 37. Output for option ICSFST2 (continued)

Element Number	Name	Description
		The fourth character in this string indicates if Duplicate Key Token Checking controls have been enabled for the PKDS. The numbers that can appear in the fourth character of this string are: **Number** — **Meaning** **0** Duplicate Key Token Checking is not enabled for the PKDS. **1** Duplicate Key Token Checking is enabled for the PKDS. **Key Store Policy is active for PKDS.** The fifth character in this string indicates if Granular Key Label Access controls have been enabled in WARN or FAIL mode. The numbers that can appear in the fifth character of this string are: **Number** — **Meaning** **0** Granular Key Label Access controls are not enabled. **1** Granular Key Label Access control is enabled in FAIL mode **2** Granular Key Label Access control is enabled in WARN mode The sixth character in this string indicates if Symmetric Key Label Export controls have been enabled for AES and/or DES keys. The numbers that can appear in the sixth character of this string are: **Number** — **Meaning** **0** Symmetric Key Label Export controls are not enabled. **1** Symmetric Key Label Export control is enabled for DES keys only. **2** Symmetric Key Label Export control is enabled for AES keys only. **3** Symmetric Key Label Export controls are enabled for both DES and AES keys.

Table 37. Output for option ICSFST2 (continued)

Element Number	Name	Description
		The seventh character in this string indicates if PKA Key Management Extensions have been enabled in either WARN or FAIL mode, and, if so, whether a SAF key ring or a PKCS #11 token is identified as the trusted certificate repository. (The trusted certificate repository is identified using the APPLDATA field of the CSF.PKAEXTNS.ENABLE profile. If no value is specified in the APPLDATA field, a PKCS #11 token is assumed.) The numbers that can appear in the seventh character of this string are: **Number** **Meaning** **0** Symmetric Key Label Export controls are not enabled. **1** PKA Key Management Extensions control is enabled in FAIL mode. The trusted certificate repository is a SAF key ring. **2** PKA Key Management Extension control is enabled in FAIL mode. The trusted certificate repository is a PKCS #11 token. **3** PKA Key Management Extensions control is enabled in WARN mode. The trusted certificate repository is a SAF key ring. **4** PKA Key Management Extension control is enabled in WARN mode. The trusted certificate repository is a PKCS #11 token.
8	ICSF Status Field 6	Status of ICSF **Number** **Meaning** **0** ICSF started **1** ICSF initialized **2** ECC master key valid, internal keys supported **3** ECC master key valid, external keys also supported
9	ICSF Status Field 7	Status of ICSF **Number** **Meaning** **0** ICSF started **1** ICSF initialized **2** RSA master key valid

Table 37. Output for option ICSFST2 (continued)

Element Number	Name	Description
10	ICSF Status Field 8	Status of ICSF **Number** **Meaning** **0** ICSF started **1** ICSF initialized **2** DES master key valid
11	ICSF Status Field 9	Status of ICSF **Number** **Meaning** **0** PKA callable services disabled. **1** PKA callable services enabled. See Usage Notes for additional information.
12	Future use	Currently blanks

Table 38. Output for option NUM-DECT

Element Number	Description
1	The number of bytes required for the output of a STATDECT request. This is the number of decimalization tables loaded times 20 bytes. This is a four-byte binary number.

Table 39. Output for option STATAES

Element Number	Name	Description
1	AES NMK Status	State of the AES new master key register: **Number** **Meaning** **1** Register is clear **2** Register contains a partially complete key **3** Register contains a complete key
2	AES CMK Status	State of the AES current master key register: **Number** **Meaning** **1** Register is clear **2** Register contains a key
3	AES OMK Status	State of the AES old master key register: **Number** **Meaning** **1** Register is clear **2** Register contains a key

Table 39. Output for option STATAES (continued)

Element Number	Name	Description
4	AES key length enablement	The maximum AES key length that is enabled by the function control vector. The value is 0 (if no AES key length is enabled in the FCV), 128, 192, or 256.

Table 40. Output for option STATCCA

Element Number	Name	Description
1	NMK Status	State of the DES New Master Key Register: **First character** **Meaning** **1** Register is clear **2** Register contains a partially complete key **3** Register contains a complete key **Last character** **Meaning (when MOREMKS keyword specified in rule array)** **blank** Register contains a 16-byte key **1** Register contains a 24-byte key
2	CMK Status	State of the DES Current Master Key Register: **First character** **Meaning** **1** Register is clear **2** Register contains a complete key **Last character** **Meaning (when MOREMKS keyword specified in rule array)** **blank** Register contains a 16-byte key **1** Register contains a 24-byte key
3	OMK Status	State of the DES Old Master Key Register: **First character** **Meaning** **1** Register is clear **2** Register contains a complete key **Last character** **Meaning (when MOREMKS keyword specified in rule array)** **blank** Register contains a 16-byte key **1** Register contains a 24-byte key
4	CCA Application Version	A character string that identifies the version of the CCA application program that is running in the coprocessor.

Table 40. Output for option STATCCA (continued)

Element Number	Name	Description
5	CCA Application Build Date	A character string containing the build date for the CCA application program that is running in the coprocessor.
6	User Role	A character string containing the Role identifier which defines the host application user's current authority.

Table 41. Output for option STATCCAE

Element Number	Name	Description
1	Symmetric NMK Status	State of the DES New Master Key Register: **First character** **Meaning** **1** Register is clear **2** Register contains a partially complete key **3** Register contains a complete key **Last character** **Meaning (when MOREMKS keyword specified in rule array)** **blank** Register contains a 16-byte key **1** Register contains a 24-byte key
2	Symmetric CMK Status	State of the DES Current Master Key Register: **First character** **Meaning** **1** Register is clear **2** Register contains a complete key **Last character** **Meaning (when MOREMKS keyword specified in rule array)** **blank** Register contains a 16-byte key **1** Register contains a 24-byte key
3	Symmetric OMK Status	State of the DES Old Master Key Register: **First character** **Meaning** **1** Register is clear **2** Register contains a complete key **Last character** **Meaning (when MOREMKS keyword specified in rule array)** **blank** Register contains a 16-byte key **1** Register contains a 24-byte key

Table 41. Output for option STATCCAE (continued)

Element Number	Name	Description
4	CCA Application Version	A character string that identifies the version of the CCA application program that is running in the coprocessor.
5	CCA Application Build Date	A character string containing the build date for the CCA application program that is running in the coprocessor.
6	User Role	A character string containing the Role identifier which defines the host application user's current authority.
7	RSA NMK Status	State of the RSA New Master Key Register: **Number** **Meaning** **1** Register is clear **2** Register contains a partially complete key **3** Register contains a complete key
8	RSA CMK Status	State of the RSA Current Master Key Register: **Number** **Meaning** **1** Register is clear **2** Register contains a key
9	RSA OMK Status	State of the RSA Old Master Key Register: **Number** **Meaning** **1** Register is clear **2** Register contains a key

Table 42. Output for option STATCARD

Element Number	Name	Description
1	Number of installed adapters	The number of active cryptographic coprocessors installed in the machine. This only includes coprocessors that have CCA software loaded (including those with CCA UDX software).
2	DES hardware level	A numeric character string containing an integer value identifying the version of DES hardware that is on the coprocessor.
3	RSA hardware level	A numeric character string containing an integer value identifying the version of RSA hardware that is on the coprocessor.
4	POST Version	A character string identifying the version of the coprocessor's Power-On Self Test (POST) firmware. The first four characters define the POST0 version and the last four characters define the POST1 version.

Table 42. Output for option STATCARD (continued)

Element Number	Name	Description
5	Coprocessor Operating System Name	A character string identifying the operating system firmware on the coprocessor. Padding characters are blanks.
6	Coprocessor Operating System Version	A character string identifying the version of the operating system firmware on the coprocessor.
7	Coprocessor Part Number	A character string containing the eight-character part number identifying the version of the coprocessor.
8	Coprocessor EC Level	A character string containing the eight-character EC (engineering change) level for this version of the coprocessor.
9	Miniboot Version	A character string identifying the version of the coprocessor's miniboot firmware. This firmware controls the loading of programs into the coprocessor. The first four characters define the MiniBoot0 version and the last four characters define the MiniBoot1 version.
10	CPU Speed	A numeric character string containing the operating speed of the microprocessor chip, in megahertz.
11	Adapter ID (Also see element number 15)	A unique identifier manufactured into the coprocessor. The coprocessor's Adapter ID is an eight-byte binary value.
12	Flash Memory Size	A numeric character string containing the size of the flash EPROM memory on the coprocessor, in 64-kilobyte increments.
13	DRAM Memory Size	A numeric character string containing the size of the dynamic RAM (DRAM) on the coprocessor, in kilobytes.
14	Battery-Backed Memory Size	A numeric character string containing the size of the battery-backed RAM on the coprocessor, in kilobytes.
15	Serial Number	A character string containing the unique serial number of the coprocessor. The serial number is factory installed and is also reported by the CLU utility in a coprocessor signed status message.

For STATDECT, the output is a table of up to 100 PIN decimalization tables as shown in the following table. The maximum size is 2000 bytes.

Table 43. Output for option STATDECT

Offset	Field	Description
0	Number	Numeric character indicating the table number
3	State	Character indicating the state of the table **L** loaded **A** active
4	Table	16-byte decimalization table

Table 44. Output for option STATDIAG

Element Number	Name	Description
1	Battery State	A numeric character string containing a value which indicates whether the battery on the coprocessor needs to be replaced: **Number** **Meaning** **1** Battery is good **2** Battery should be replaced
2	Intrusion Latch State	A numeric character string containing a value which indicates whether the intrusion latch on the coprocessor is set or cleared: **Number** **Meaning** **1** Latch is cleared **2** Latch is set
3	Error Log Status	A numeric character string containing a value which indicates whether there is data in the coprocessor CCA error log. **Number** **Meaning** **1** Error log is empty **2** Error log contains data but is not yet full **3** Error log is full
4	Mesh Intrusion	A numeric character string containing a value to indicate whether the coprocessor has detected tampering with the protective mesh that surrounds the secure module — indicating a probable attempt to physically penetrate the module. **Number** **Meaning** **1** No intrusion detected **2** Intrusion attempt detected.
5	Low Voltage Detected	A numeric character string containing a value to indicate whether a power supply voltage was under the minimum acceptable level. This may indicate an attempt to attack the security module. **Number** **Meaning** **1** Only acceptable voltages have been detected **2** A voltage has been detected under the low-voltage tamper threshold

Table 44. Output for option STATDIAG (continued)

Element Number	Name	Description
6	High Voltage Detected	A numeric character string containing a value to indicate whether a power supply voltage was higher than the maximum acceptable level. This may indicate an attempt to attack the security module. **Number** **Meaning** **1** Only acceptable voltages have been detected **2** A voltage has been detected that is higher than the high-voltage tamper threshold
7	Temperature Range Exceeded	A numeric character string containing a value to indicate whether the temperature in the secure module was outside of the acceptable limits. This may indicate an attempt to obtain information from the module: **Number** **Meaning** **1** Temperature is acceptable **2** Detected temperature is outside an acceptable limit
8	Radiation Detected	A numeric character string containing a value to indicate whether radiation was detected inside the secure module. This may indicate an attempt to obtain information from the module: **Number** **Meaning** **1** No radiation has been detected **2** Radiation has been detected
9, 11, 13, 15, 17	Last Five Commands Run	These five rule-array elements contain the last five commands that were executed by the coprocessor CCA application. They are in chronological order, with the most recent command in element 9. Each element contains the security API command code in the first four characters and the subcommand code in the last four characters.
10, 12, 14,16, 18	Last Five Return Codes	These five rule-array elements contain the SAPI return codes and reason codes corresponding to the five commands in rule-array elements 9, 11, 13, 15, and 17. l Each element contains the return code in the first four characters and the reason code in the last four characters.

Table 45. Output for option STATEID

Element Number	Name	Description
1	EID	During initialization, a value of zero is set in the coprocessor.

Table 46. Output for option STATEXPT

Element Number	Name	Description
1	Base CCA Services Availability	A numeric character string containing a value to indicate whether base CCA services are available. **Number** **Meaning** **0** Base CCA services are not available **1** Base CCA services are available
2	CDMF Availability	A numeric character string containing a value to indicate whether CDMF is available. **Number** **Meaning** **0** CDMF encryption is not available
3	56-bit DES Availability	A numeric character string containing a value to indicate whether 56-bit DES encryption is available. **Number** **Meaning** **0** 56-bit DES encryption is not available **1** 56-bit DES encryption is available
4	Triple-DES Availability	A numeric character string containing a value to indicate whether triple-DES encryption is available. **Number** **Meaning** **0** Triple-DES encryption is not available **1** Triple-DES encryption is available
5	SET Services Availability	A numeric character string containing a value to indicate whether SET (Secure Electronic Transaction) services are available. **Number** **Meaning** **0** SET Services are not available **1** SET Services are available

Table 46. Output for option STATEXPT (continued)

Element Number	Name	Description
6	Maximum Modulus for Symmetric Key Encryption	A numeric character string containing the maximum modulus size that is enabled for the encryption of symmetric keys. This defines the longest public-key modulus that can be used for key management of symmetric-algorithm keys. **Number** **Meaning** **0** RSA not available **1024** RSA 1024 key size **2048** RSA 2048 key size **4096** RSA 4096 key size

Table 47. Output for option STATAPKA

Element Number	Name	Description
1	ECC NMK status	The state of the ECC new master key register: **Number** **Meaning** **1** Register is clear. **2** Register contains a partially complete key. **3** Register contains a complete key.
2	ECC CMK status	The state of the ECC current master key register: **Number** **Meaning** **1** Register is clear. **2** Register contains a key.
3	ECC OMK status	The state of the ECC old master key register: **Number** **Meaning** **1** Register is clear. **2** Register contains a key.
4	ECC key length enablement	The maximum ECC curve size that is enabled by the function control vector. The value will be 0 (if no ECC keys are enabled in the FCV) and 521 for the maximum size.

Table 48. Output for option WRAPMTHD

Element Number	Name	Description
1	Internal tokens	Default wrapping method for internal tokens. **Number** **Meaning** **0** Keys will be wrapped with the original method **1** Keys will be wrapped with the enhanced X9.24 method
2	External tokens	Default wrapping method for external tokens. **Number** **Meaning** **0** Keys will be wrapped with the original method **1** Keys will be wrapped with the enhanced X9.24 method

Table 49. Output for option STATP11

Element Number	Name	Description
1	P11 NMK Status	State of the P11 new master key register: **Number** **Meaning** **1** Register is clear **2** Register contains an uncommitted key **3** Register contains a committed key
2	P11 CMK	Status State of the P11 current master key register: **Number** **Meaning** **1** Register is clear **2** Register contains a key
3	Compliance Mode	Current compliance mode for the coprocessor. An 8-byte hexadecimal number that is the sum of the active compliance modes: **Number** **Meaning** **n** An 8-byte hexadecimal number that is the sum of the active compliance modes: • 1 - FIPS 2009 • 2 - BSI 2009 • 4 - FIPS 2011 • 8 - BSI 2011

Table 49. Output for option STATP11 (continued)

Element Number	Name	Description
4	Firmware version	Coprocessor PKCS #11 firmware version number as an 8-byte hexadecimal value.
5	Serial Number	A character string containing the unique serial number of the coprocessor. The serial number is factory installed.
6 – 12	Future use	Currently blanks

Table 50. Output for option SIZEWPIN

Description
The number of bytes of storage required for the output of a STATWPIN request. The value is a 4-byte binary number.

For STATWPIN, the output is a table of up to 20 weak PINs. Each entry in the table is formatted as shown in the following table. The maximum size is 460 bytes. The data in the table in character format (EBCDIC).

Table 51. Output for option STATWPIN

Offset	Length	Description
0	1	Weak PIN structure type
1	3	Numeric character indicating the table number
4	1	Character indicating the state of the table: **Character** **Meaning** **A** Active **L** Loaded
5	2	PIN length
7	4 to 16	Weak PIN

reserved_data_length

Direction	Type
Input	Integer

The length of the *reserved_data* parameter. The value must be 0.

reserved_data

Direction	Type
Input	String

This field is not used.

Usage Notes

RACF will be invoked to check authorization to use this service.

PKA key generate available indicates the PKA callable services are enabled and there is at least one ACTIVE coprocessor.

The options ICSFSTAT and ICSFST2 report on the state of PKA callable services. ICSFSTAT reports it in element 2. ICSFST2 reports it in elements 3 and 11. There is a subtle difference between the three options. ICSFSTAT reports PKA callable services as enabled only after the DES master key is loaded and valid. ICSFSTAT does not report PKA callable services as enabled when only the AES master key is loaded and valid. Option ICSFST2 element 3 reports PKA callable services as enabled when the DES and/or AES master key is loaded and valid. Option ICSFST2 element 11 reports PKA callable services as enabled when neither the DES nor AES master keys are loaded and valid.

Note: If your system has CEX3C or later coprocessors, the PKA callable services control may not be available. The PKA callable services state will be the same as the RSA master key. If the RSA master key is active, the PKA callable services will be enabled in the ICSFSTAT and ICSFST2 reports.

Required Hardware

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

Table 52. ICSF Query Service required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890	None.	
IBM System z9 EC IBM System z9 BC	None.	
IBM System z10 EC IBM System z10 BC	None.	
IBM zEnterprise 196 IBM zEnterprise 114	None.	
IBM zEnterprise EC12 IBM zEnterprise BC12	None	

ICSF and TSS Return and Reason Codes

This topic provides the return and reason codes related to the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods:

- "Reason Codes for Return Code 0 (0)"
- "Reason Codes for Return Code 8 (8)" on page 121

Reason Codes for Return Code 0 (0)

Table 53. Reason Codes for Return Code 0 (0)

Reason Code Hex (Decimal)	Description
87D (2173)	The call to the callable service was successfully processed. The key token format was already payload version 1 (fixed-length).

Reason Codes for Return Code 8 (8)

Table 54. Reason Codes for Return Code 8 (8)

Reason Code Hex (Decimal)	Description
87E (2174)	The provided data is not hexadecimal digits. **User action**: Provide the data in the correct format.
87F (2175)	A weak PIN was presented. The PIN change has been rejected. **User action**: Provide another PIN.
87E (2177)	The PAN presented to the DK PAN change service was the same as the PAN in the encrypted PIN block. The change has been rejected. **User action**: Check the PAN parameters and correct the parameter is error.
882 (2178)	The PAN data supplied to the DK Deterministic PIN Generate service does not match the supplied data in the *account_info_ER* parameter. **User action**: Supply the correct PAN.
895 (2197)	The input PIN could not be verified. **User action**: Ensure that the correct values were supplied for the parameters used to verify the PIN and ensure that the input PIN is correct.
896 (2198)	The supplied MAC was compared against a MAC calculated from the supplied parameters. The MACs did not match. **User action**: Ensure that the correct values were supplied for the parameters used to calculate the MAC and ensure that the supplied MAC is correct.
897 (2199)	A variable-length symmetric key-token (version X'05') contains invalid key-usage field data. **User action**: Supply a valid key token.
899 (2201)	A variable-length symmetric key-token (version X'05') contains invalid key-management field data. **User action**: Supply a valid key token.
B21 (2849)	A keyword was passed in the *service_data* parameter of Key Token Build2 service and it is not a valid keyword for the service. **User action**: Correct the keywords in the *service_data* parameter.
B22 (2850)	The combination of keywords in the *service_data* parameter of the Key Token Build2 service is not valid. **User action**: Check the keywords allowed for the key type being derived and correct the *service_data* parameter.
B23 (2851)	The *service_data_length* parameter of the Key Token Build2 service does not have a valid value. **User action**: The length must be a multiple of 8 and the keywords in the *service_data* parameter must be left-justified and padded with blanks.
B81 (2945)	A required keyword for the key type being derived is not in the *service_data* parameter of the Key Token Build2 service. **User action**: Review the keywords for the key type being derived and supply all required keywords.

Key Token Formats

Variable-length Symmetric Key Token

The following table presents the format for a variable-length symmetric key token. The length of the token depends on the key type and algorithm.

Table 55. Variable-length Symmetric Key Token

Offset (Dec)	Length of Field (Bytes)	Description
		Header
0	1	Token flag **X'00'** for null tokens **X'01'** for internal tokens **X'02'** for external tokens
1	1	Reserved (X'00')
2	2	Length of the token in bytes
4	1	Token version number X'05' (May be X'00' for null tokens)
5	3	Reserved (X'000000')
		Wrapping information
8	1	Key material state. **X'00'** no key present (internal or external) **X'01'** key is clear (internal) **X'02'** key is encrypted under a key-encrypting key (external) **X'03'** key is encrypted under the master key (internal)
9	1	Key verification pattern (KVP) type. **X'00'** No KVP **X'01'** AES master key verification pattern **X'02'** key-encrypting key verification pattern
10	16	Verification pattern of the key used to wrap the payload. Value is left justified.
26	1	Wrapping method - This value indicates the wrapping method used to protect the data in the encrypted section. **X'00'** key is in the clear **X'02'** AESKW **X'03'** PKOAEP2
27	1	Hash algorithm used in wrapping algorithm. • For wrapping method X'00' **X'00'** None. For clear key tokens. • For wrapping method X'02' **X'02'** SHA-256 • For wrapping method X'03' **X'01'** SHA-1 **X'02'** SHA-256 **X'04'** SHA-384 **X'08'** SHA-512

Table 55. Variable-length Symmetric Key Token (continued)

Offset (Dec)	Length of Field (Bytes)	Description
28	1	Payload version **X'00'** Variable-length payload **X'01'** Fixed-length payload All other values are reserved and must not be used.
29	1	Reserved (X'00')
		AESKW Components: Associated data and clear key or encrypted AESKW payload
		Associated data section
30	1	Associated data version (X'01')
31	1	Reserved (X'00')
32	2	Length of the associated data in bytes: *adl*
34	1	Length of the key name in bytes: *kl*
35	1	Length of the IBM extended associated data in bytes: *iead*
36	1	Length of the installation-definable associated data in bytes: *uad*
37	1	Reserved (X'00')
38	2	Length of the payload in bits: *pl*
40	1	Reserved (X'00')
41	1	Type of algorithm for which the key can be used **X'01'** DES **X'02'** AES **X'03'** HMAC
42	2	Key type: For algorithm AES: **X'0001'** CIPHER **X'0002'** MAC **X'0003'** EXPORTER **X'0004'** IMPORTER **X'0005'** PINPROT **X'0006'** PINCALC **X'0007'** PINPRW **X'0009'** DKYGENKY For algorithm HMAC: **X'0002'** MAC For algorithm DES: **X'0008'** DESUSECV
44	1	Key-usage field count (*kuf*) - (1 byte) Key-usage field information defines restrictions on the use of the key.

Table 55. Variable-length Symmetric Key Token (continued)

Offset (Dec)	Length of Field (Bytes)	Description
45	*kuf* * 2	Key-usage fields (*kuf* * 2 bytes) • For HMAC algorithm keys, refer to Table 62 on page 133. • For AES algorithm Key-Encrypting keys (Exporter or Importer), refer to Table 63 on page 134. • For AES algorithm Cipher keys, refer to Table 64 on page 137. • For AES algorithm MAC keys, refer to Table 56 on page 125. • For AES algorithm PINCALC keys, refer to Table 57 on page 126. • For AES algorithm PINPROT keys, refer to Table 58 on page 127. • For AES algorithm PINPRW keys, refer to Table 59 on page 128. • For AES algorithm DKYGENKY keys, refer to Table 60 on page 130. • For DESUSECV keys, refer to Table 61 on page 132
45 + *kuf* * 2	1	Key-management field count (*kmf*) - (2 byte): • For AES and HMAC keys: 2 (no pedigree information) or 3 (has pedigree information) • For DESUSECV keys: 1 Key-management field information describes how the data is to be managed or helps with management of the key material.
46 + *kuf* * 2	*kuf* * 2	Key-management fields (kmf * 2 bytes): • For AES and HMAC algorithm keys, refer to Table 65 on page 139 • For DESUSECV keys, refer to Table 66 on page 142
46 + *kuf* * 2 + *kmf* * 2	*kl*	Key name
46 + *kuf* * 2 + *kmf* * 2 + *kl*	*iead*	IBM extended associated data
46 + *kuf* * 2 + *kmf* * 2 + *kl* + *iead*	*uad*	Installation-defined associated data
		Clear key or encrypted payload
30 + *adl*	*(pl+7)/8*	**Encrypted AESKW payload (internal keys)**: The encrypted AESKW payload is created from the unencrypted AESKW payload which is made up of the ICV/pad length/hash options and hash length/hash options/hash of the associated data/key material/padding. See unencrypted AESKW payload below. **Encrypted PKOAEP2 payload (external keys)**: The encrypted PKOAEP2 payload is created using the PKCS #1 v1.2 encoding method for a given hash algorithm. The message (M) inside the encoding contains: [2 bytes: bit length of key] || [clear HMAC key]. M is encoded using OAEP and then encrypted with an RSA public key according to the standard. **Clear key payload**: When the key is clear, only the key material will be in the payload padded to the nearest byte with binary zeros.
30 + *adl* + (pl+7)/8		End of AESKW components

Table 56. AES Algorithm MAC Key Associated Data

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (kuf): 2 – 3 Count is based on whether the key is DK enabled or not: *kuf* **DK enabled** **2** No **3** Yes
45	2	Key-usage field 1 High-order byte: **B'00xx xxxx'** Undefined. **B'01xx xxxx'** Key cannot be used for generate; key can be used for verify. **B'10xx xxxx'** Key can be used for generate; key cannot be used for verify. **B'11xx xxx*'** Key can be used for generate and verify. Not valid if offset 50 is X'01'. All unused bits are reserved and must be zero. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where *uuu* are UDX-defined bits. All unused bits are reserved and must be zero.
47	2	Key-usage field 2 High-order byte: **X'01'** CMAC mode. All unused bits are reserved and must be zero. Low-order byte: All bits are reserved and must be zero.

Table 56. AES Algorithm MAC Key Associated Data (continued)

Offset (Dec)	Length of Field (Bytes)	Description
49	2	Key-usage field 3 High-order byte when DK enabled: **X'01'** PIN_OP (DKPINOP) **X'03'** PIN_ADMIN1 (DKPINAD1) **X'04'** PIN_ADMIN2 (DKPINAD2) All unused values are reserved and must not be used. Low-order byte: **X'01'** DK enabled. All unused values are reserved and must not be used.

Table 57. AES Algorithm PINCALC Key Associated Data

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (*kuf*): 3
45	2	Key-usage field 1 High-order byte: **B'00xx xxxx'** Undefined. **B'10xx xxxx'** Key can be used for generate; key cannot be used for verify. All unused bits are reserved and must be zero. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where *uuu* are UDX-defined bits. All unused bits are reserved and must be zero.
47	2	Key-usage field 2 High-order byte: **X'00'** Key can be used for Cipher Block Chaining (CBC). All unused values are reserved and must not be used. Low-order byte: All bits are reserved and must be zero.

Table 57. AES Algorithm PINCALC Key Associated Data (continued)

Offset (Dec)	Length of Field (Bytes)	Description
49	2	Key-usage field 3 High-order byte when DK enabled: **X'01'** PIN_OP (DKPINOP) All unused values are reserved and must not be used. Low-order byte: **X'01'** DK enabled. All unused values are reserved and must not be used.

Table 58. AES Algorithm PINPROT Key Associated Data

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (kuf): 3
45	2	Key-usage field 1 High-order byte: **B'00xx xxxx'** Undefined. **B'01xx xxxx'** Key cannot be used for encryption; key can be used for decryption. **B'10xx xxxx'** Key can be used for encryption; key cannot be used for decryption. **B'11xx xxxx'** Undefined. All unused bits are reserved and must be zero. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where *uuu* are UDX-defined bits. All unused bits are reserved and must be zero.

Table 58. AES Algorithm PINPROT Key Associated Data (continued)

Offset (Dec)	Length of Field (Bytes)	Description
47	2	Key-usage field 2 High-order byte: **X'00'** Key can be used for Cipher Block Chaining (CBC). All unused values are reserved and must not be used. Low-order byte: All bits are reserved and must be zero.
49	2	Key-usage field 3 High-order byte when DK enabled: **X'01'** PIN_OP (DKPINOP) **X'02'** PIN_OPP (DKPINOPP) **X'03'** PIN_ADMIN1 (DKPINAD1) All unused values are reserved and must not be used. Low-order byte: **X'01'** DK enabled. All unused values are reserved and must not be used.

Table 59. AES Algorithm PINPRW Key Associated Data

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (kuf): 3

Table 59. AES Algorithm PINPRW Key Associated Data (continued)

Offset (Dec)	Length of Field (Bytes)	Description
45	2	Key-usage field 1 High-order byte: **B'00xx xxxx'** Undefined. **B'01xx xxxx'** Key cannot be used for generate; key can be used for verify. **B'10xx xxxx'** Key can be used for generate; key cannot be used for verify. **B'11xx xxxx'** Undefined. All unused bits are reserved and must be zero. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where *uuu* are UDX-defined bits. All unused bits are reserved and must be zero.
47	2	Key-usage field 2 High-order byte: **X'01'** CMAC mode All unused values are reserved and must not be used. Low-order byte: All bits are reserved and must be zero.
49	2	Key-usage field 3 High-order byte when DK enabled: **X'01'** PIN_OP (DKPINOP) All unused values are reserved and must not be used. Low-order byte: **X'01'** DK enabled. All unused values are reserved and must not be used.

Table 60. AES Algorithm DKYGENKY Key Associated Data

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (kuf): 2, 4, 5, or 6
45	2	Key-usage field 1 High-order byte: Defines the key type to be generated. **X'00'** Any type listed below (D-ALL) **X'01'** CIPHER (D-CIPHER) **X'02'** MAC (D-MAC) **X'03'** EXPORTER (D-EXP) **X'04'** IMPORTER (D-IMP) **X'05'** PINPROT (D-PPROT) **X'06'** PINCALC (D-PCALC) **X'07'** PINPRW (D-PPRW) All other values are reserved and undefined. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where *uuu* are UDX-defined bits. All unused bits are reserved and must be zero.

Table 60. AES Algorithm DKYGENKY Key Associated Data (continued)

Offset (Dec)	Length of Field (Bytes)	Description
47	2	Key-usage field 2: Indicates the key usage. High-order byte (key-usage field level of control): **B'1xxx xxxx'** The key usage fields of the key to be generated must be equal (KUF-MBE) to the related generated key usage fields that start with key usage field 3 below. **B'0xxx xxxx'** The key usage fields of the key identifier to be generated must be permitted (KUF-MBP) based on the related generated-key usage fields that start with key usage field 3 below. A key to be diversified is not permitted to have a higher level of usage than the related key usage fields permit. The key to be diversified is only permitted to have key usage that is less than or equal to the related key usage fields. The UDX-ONLY bit of the related key usage fields must always be equal in both the generating key and the generated key. Undefined when the value at offset 45 = X'00' (D-ALL). All other values are reserved and undefined. Low-order byte (key-derivation sequence level): **X'00'** DKYL0. Generate a key based on the key usage byte at offset 45. All other values are reserved and undefined.
49 (if defined)	2	Key-usage field 3 (related generated key usage fields): These values determine allowable key usage of key to be generated. Meaning depends on value of offset 45: **X'01'** Same as key-usage field 1 of AES CIPHER key. **X'02'** Same as key-usage field 1 of AES MAC key. **X'03'** Same as key-usage field 1 of AES EXPORTER key. **X'04'** Same as key-usage field 1 of AES IMPORTER key. **X'05'** Same as key-usage field 1 of AES PINPROT key. **X'06'** Same as key-usage field 1 of AES PINCALC key. **X'07'** Same as key-usage field 1 of AES PINPRW key.

Table 60. AES Algorithm DKYGENKY Key Associated Data (continued)

Offset (Dec)	Length of Field (Bytes)	Description
51 (if defined)	2	Key-usage field 4 (related generated key usage fields): These values determine allowable key usage of key to be generated. Meaning depends on value of offset 45: **X'01'** Same as key-usage field 2 of AES CIPHER key. **X'02'** Same as key-usage field 2 of AES MAC key. **X'03'** Same as key-usage field 2 of AES EXPORTER key. **X'04'** Same as key-usage field 2 of AES IMPORTER key. **X'05'** Same as key-usage field 2 of AES PINPROT key. **X'06'** Same as key-usage field 2 of AES PINCALC key. **X'07'** Same as key-usage field 2 of AES PINPRW key.
53 (if defined)	2	Key-usage field 5 (related generated key usage fields): These values determine allowable key usage of key to be generated. Meaning depends on value of offset 45: **X'02'** Same as key-usage field 3 of AES MAC key. **X'03'** Same as key-usage field 3 of AES EXPORTER key. **X'04'** Same as key-usage field 3 of AES IMPORTER key. **X'05'** Same as key-usage field 3 of AES PINPROT key. **X'06'** Same as key-usage field 3 of AES PINCALC key. **X'07'** Same as key-usage field 3 of AES PINPRW key.
55 (if defined)	2	Key-usage field 6 (related generated key usage fields): These values determine allowable key usage of key to be generated. Meaning depends on value of offset 45: **X'03'** Same as key-usage field 4 of AES EXPORTER key. **X'04'** Same as key-usage field 4 of AES IMPORTER key.

Table 61. DESUSECV Key-usage fields

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (*kuf*): 1

Table 61. DESUSECV Key-usage fields (continued)

Offset (Dec)	Length of Field (Bytes)	Description
45	2	Key-usage field 1 High-order byte: **B'0000 0000'** Reserved All unused bits are reserved and must be zero. Low-order byte: **B'0000 0000'** Reserved All unused bits are reserved and must be zero.

Table 62. HMAC Algorithm Key-usage fields

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (*kuf*): 2
45	2	Key-usage field 1 High-order byte: **1xxx xxxx** Key can be used for generate. **x1xx xxxx** Key can be used for verify. All unused bits are reserved and must be zero. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where uuu are UDX-defined bits. All unused bits are reserved and must be zero.

Table 62. HMAC Algorithm Key-usage fields (continued)

Offset (Dec)	Length of Field (Bytes)	Description
47	2	Key-usage field 2 High-order byte: **1xxx xxxx** SHA-1 hash method is allowed for the key. **x1xx xxxx** SHA-224 hash method is allowed for the key. **xx1x xxxx** SHA-256 hash method is allowed for the key. **xxx1 xxxx** SHA-384 hash method is allowed for the key. **xxxx 1xxx** SHA-512 hash method is allowed for the key. All unused bits are reserved and must be zero. Low-order byte: All bits are reserved and must be zero.

Table 63. AES Algorithm KEK Key-usage fields

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (*kuf*): 4

Table 63. AES Algorithm KEK Key-usage fields (continued)

Offset (Dec)	Length of Field (Bytes)	Description
45	2	Key-usage field 1 High-order byte for EXPORTER: **1xxx xxxx** Key can be used for EXPORT. **x1xx xxxx** Key can be used for TRANSLAT. **xx1x xxxx** Key can be used for GENERATE-OPEX. **xxx1 xxxx** Key can be used for GENERATE-IMEX. **xxxx 1xxx** Key can be used for GENERATE-EXEX. **xxxx x1xx** Key can be used for GENERATE-PUB. All unused bits are reserved and must be zero. High-order byte for IMPORTER: **1xxx xxxx** Key can be used for IMPORT. **x1xx xxxx** Key can be used for TRANSLAT. **xx1x xxxx** Key can be used for GENERATE-OPIM. **xxx1 xxxx** Key can be used for GENERATE-IMEX. **xxxx 1xxx** Key can be used for GENERATE-IMIM. **xxxx x1xx** Key can be used for GENERATE-PUB. All unused bits are reserved and must be zero. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where uuu are UDX-defined bits. All unused bits are reserved and must be zero.

Table 63. AES Algorithm KEK Key-usage fields (continued)

Offset (Dec)	Length of Field (Bytes)	Description
47	2	Key-usage field 2 High-order byte: **1xxx xxxx** Key can wrap a TR-31 key. All unused bits are reserved and must be zero. Low-order byte: **xxxx xxx1** This KEK can export a key in RAW format. All unused bits are reserved and must be zero
49	2	Key-usage field 3 High-order byte: **1xxx xxxx** Key can wrap DES keys **x1xx xxxx** Key can wrap AES keys **xx1x xxxx** Key can wrap HMAC keys **xxx1 xxxx** Key can wrap RSA keys **xxxx 1xxx** Key can wrap ECC keys All unused bits are reserved and must be zero. Low-order byte: All bits are reserved and must be zero.

Table 63. AES Algorithm KEK Key-usage fields (continued)

Offset (Dec)	Length of Field (Bytes)	Description
51	2	Key-usage field 4 High-order byte: **1xxx xxxx** Key can wrap DATA class keys **x1xx xxxx** Key can wrap KEK class keys **xx1x xxxx** Key can wrap PIN class keys **xxx1 xxxx** Key can wrap DERIVATION class keys **xxxx 1xxx** Key can wrap CARD class keys **xxxx x1xx** Key can wrap CVAR class keys All unused bits are reserved and must be zero. Low-order byte: All bits are reserved and must be zero.

Table 64. AES Algorithm Cipher Key Associated Data

Offset (Dec)	Length of Field (Bytes)	Description
44	1	Key-usage field count (*kuf*): 2

Table 64. AES Algorithm Cipher Key Associated Data (continued)

Offset (Dec)	Length of Field (Bytes)	Description
45	2	Key-usage field 1 High-order byte: **1xxx xxxx** Key can be used for encryption. **x1xx xxxx** Key can be used for decryption. **xx1x xxxx** Key can be used for cipher text translate only. All unused bits are reserved and must be zero. Low-order byte: **xxxx 1xxx** The key can only be used in UDXs (used in KGN, KIM, KEX). **xxxx 0xxx** The key can be used in both UDXs and CCA. **xxxx xuuu** Reserved for UDXs, where uuu are UDX-defined bits. All unused bits are reserved and must be zero.
47	2	Key-usage field 2 High-order byte: **X'00'** Key can be used for Cipher Block Chaining (CBC). **X'01'** Key can be used for Electronic Code Book (ECB). **X'02'** Key can be used for Cipher Feedback (CFB). **X'03'** Key can be used for Output Feedback (OFB). **X'04'** Key can be used for Galois/Counter Mode (GCM) **X'05'** Key can be used for XEX-based Tweaked CodeBook Mode with CipherText Stealing (XTS) All unused values are reserved and must not be used. Low-order byte: All bits are reserved and must be zero.

Table 65. AES and HMAC algorithm key-management fields

Offset (Dec)	Length of Field (Bytes)	Description
48	2	Key-management field 1 High-order byte: **1xxx xxxx** Allow export using symmetric key. **x1xx xxxx** Allow export using unauthenticated asymmetric key. **xx1x xxxx** Allow export using authenticated asymmetric key. **xxx1 xxxx** Allow export in RAW format. All other bits are reserved and must be zero. Low-order byte: --symmetric-- **1xxx xxxx** Prohibit export using DES key. **x1xx xxxx** Prohibit export using AES key. --asymmetric-- **xxxx 1xxx** Prohibit export using RSA key. All other bits are reserved and must be zero.

Table 65. AES and HMAC algorithm key-management fields (continued)

Offset (Dec)	Length of Field (Bytes)	Description
48 + *kuf* * 2	2	Key-management field 2 High-order byte: **11xx xxxx** Key, if present, is incomplete. Key requires at least 2 more parts. **10xx xxxx** Key, if present, is incomplete. Key requires at least 1 more part. **01xx xxxx** Key, if present, is incomplete. Key can be completed or have more parts added. **00xx xxxx** Key, if present, is complete. No more parts can be added. All other bits are reserved and must be zero. Low-order byte (Security History): **xxx1 xxxx** Key was encrypted with an untrusted KEK. **xxxx 1xxx** Key was in a format without type/usage attributes. **xxxx x1xx** Key was encrypted with key weaker than itself. **xxxx xx1x** Key was in a non-CCA format. **xxxx xxx1** Key was encrypted in ECB mode. All other bits are reserved and must be zero.
50 + *kuf* * 2	2	Key-management field 3 - Pedigree (this field may or may not be present) Indicates how key was originally created and how it got into the current system. High-order byte: Pedigree Original **X'00'** Unknown (Key Token Build2, Key Translate2) **X'01'** Other - method other than those defined here, probably used in UDX **X'02'** Randomly Generated (Key Generate2) **X'03'** Established by key agreement (ECC Diffie-Hellman) **X'04'** Created from cleartext key components (Key Part Import2) **X'05'** Entered as a cleartext key value (Key Part Import2, Secure Key Import2) **X'06'** Derived from another key **X'07'** Cleartext keys or key parts that were entered at TKE and secured from there to the target card (operational key load) All unused values are reserved and undefined.

Table 65. AES and HMAC algorithm key-management fields (continued)

Offset (Dec)	Length of Field (Bytes)	Description
50 + *kuf* * 2 (cont'd)	2 (cont'd)	**X'00'** Unknown (Key Token Build2) **X'01'** Other - method other than those defined here, probably used in UDX **X'02'** Randomly Generated (Key Generate2) **X'03'** Established by key agreement (ECC Diffie-Hellman) **X'04'** Created from cleartext key components (Key Part Import2) **X'05'** Entered as a cleartext key value (Key Part Import2, Secure Key Import2) **X'06'** Derived from another key **X'07'** Imported from a CCA 05 variable length token with pedigree field (Symmetric Key Import2) **X'08'** Imported from a CCA 05 variable length token with no pedigree field (Symmetric Key Import2) **X'09'** Imported from a CCA token that had a CV **X'0A'** Imported from a CCA token that had no CV or a zero CV **X'0B'** Imported from a TR-31 key block that contained a CCA CV (ATTR-CV option) (TR-31 Import) **X'0C'** Imported from a TR-31 key block that did not contain a CCA CV (TR-31 Import) **X'0D'** Imported using PKCS 1.2 RSA encryption (Symmetric Key Import2) **X'0E'** Imported using PKCS OAEP encryption (Symmetric Key Import2) **X'0F'** Imported using PKA92 RSA encryption (Symmetric Key Import2) **X'10'** Imported using RSA ZERO-PAD encryption (Symmetric Key Import2) **X'11'** Converted from a CCA token that had a CV (Key Translate2) **X'12'** Converted from a CCA token that had no CV or a zero CV (Key Translate2) **X'13'** Cleartext keys or key parts that were entered at TKE and secured from there to the target card (operational key load) **X'14'** Exported from a CCA 05 variable length token with pedigree field (Symmetric Key Export) **X'15'** Exported from a CCA 05 variable length token with no pedigree field (Symmetric Key Export) **X'16'** Exported using PKCS OAEP encryption (Symmetric Key Export) All unused values are reserved and undefined.

Table 66. DESUSECV key-management fields

Offset (Dec)	Length of Field (Bytes)	Description
47	1	Key-management field count (*kmf*): 1
48	2	Key-management field 1 High-order byte: **B'0000 0000'** Reserved All unused bits are reserved and must be zero. Low-order byte: **B'0000 0000'** Reserved All unused bits are reserved and must be zero.

Access Control Points and Callable Services

Access to callable services that are executed on a coprocessor is through Access Control Points in the domain role. To execute services on the coprocessor, access control points must be enabled for each service in the domain role. The access control points available depend on the coprocessor you are using.

The TKE workstation allows you to enable or disable access control points. For systems that do not use the optional TKE Workstation, most access control points (current and new) are enabled in the domain role with the appropriate licensed internal code on the coprocessor. The table of access control points lists the default setting of each access control point.

New TKE users and non-TKE users have the default set of access control points enabled. For existing TKE users who have changed the setting of any access control point, any new access control points will not be enabled.

Note: Access control points for ICSF utilities are listed in *z/OS Cryptographic Services ICSF Administrator's Guide*.

If an access control point is disabled, the corresponding ICSF callable service will fail during execution with an access denied error.

The following tables list usage information using the following abbreviations:

AE Always enabled, can not be disabled.

ED Enabled by default.

DD Disabled by default.

SC Usage of this access control point requires special consideration.

Table 67. Access control points – Callable Services

Name	Callable Service	Usage
Diversified Key Generate2 – AES EMV1 SESS	CSNBDKG2 / CSNEDKG2	ED
Diversified Key Generate2 - DALL	CSNBDKG2 / CSNEDKG2	DD, SC

Table 67. Access control points – Callable Services (continued)

Name	Callable Service	Usage
DK PAN Modify in Transaction	CSNBDPMT / CSNEDPMT	DD
DK PIN Verify	CSNBDPV / CSNEDPV	DD
DK PIN Change	CSNBDPC / CSNEDPC	DD
DK Random PIN Generate	CSNBDRPG / CSNEDRPG	DD
Key Generate2 – DK PIN admin1 key MAC	CSNBKGN2 / CSNEKGN2	DD
Key Generate2 – DK PIN admin1 key PINPROT	CSNBKGN2 / CSNEKGN2	DD
Key Generate2 – DK PIN admin2 key MAC	CSNBKGN2 / CSNEKGN2	DD
Key Generate2 – DK PIN key set	CSNBKGN2 / CSNEKGN2	DD
Key Generate2 – DK PIN print key	CSNBKGN2 / CSNEKGN2	DD
Key Translate2 - Translate fixed to variable payload	CSNBKTR2 / CSNEKTR2	DD, SC

Chapter 3. Update of z/OS Cryptographic Services ICSF Administrator's Guide, SC14-7506-00, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF Administrator's Guide*, SC14-7506-00, for the DK AES PIN support provided by this APAR. Refer to this source document if background information is needed.

Setting up profiles in the CSFSERV general resource class

This topic provides the resource names for the new callable services that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods:

Table 68. Resource names for ICSF Callable Services

Resource Name	Callable Service Name(s)	Callable Service Description
CSNDKG2	CSNBDKG2 CSNEDKG2	Diversified Key Generate2
CSFDPC	CSNBDPC CSNEDPC	DK PIN Change
CSFDPMT	CSNBDPMT CSNEDPMT	DK PAN Modify in Transaction
CSFDPV	CSNBDPV CSNEDPV	DK PIN Verify
CSFDRPG	CSNBDRPG CSNEDRPG	DK Random PIN Generate

Managing Cryptographic Keys Using the Key Generator Utility Program

This topic provides the updated KGUP details that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods.

Using KGUP control statements

You use control statements to specify the function you want the key generator utility program (KGUP) to perform. You use job control language (JCL) to submit the control statements to KGUP. You can create and submit KGUP control statements either on your own or using the KGUP panels. OPKYLOAD control statements can not be created using the KGUP panels.

You specify information to KGUP using an ADD, UPDATE, DELETE, RENAME, SET or OPKYLOAD control statement. You use keywords on the control statement to specify:

- The function KGUP performs
- Information about the key that KGUP processes

For example, if you specify the KEY keyword on an ADD control statement, you supply a key which KGUP adds to the CKDS in an entry.

This topic describes the syntax of the control statements with their keywords. Use these rules when interpreting the syntax of the control statements:

- Specify uppercase letters and special characters as shown in the examples.
- Lowercase letters represent keyword values that you must specify.
- A bar (|) indicates a choice (OR).
- Ellipses (...) indicates that multiple entries are possible.
- Braces ({ }) denote choices, one of which you must specify.
- Brackets ([ ]) denote choices, one of which you may specify.

Control statements in the Control Statement Input data set may not be longer than 71 characters including the continuation character.

General Rules for CKDS Records

There are some general rules for creating labels for CKDS key records.

- Each label can consist of up to 64 characters. The first character must be alphabetic or a national character (#, $, @). The remaining characters can be alphanumeric, a national character (#, $, @), or a period (.).
- Labels must be unique for all key types except EXPORTER, IMPORTER, IPINENC, PINGEN, PINVER, and OPINENC
- Labels must be unique for any key record, including transport and PIN keys, created or updated using the dynamic CKDS update services.

KGUP and the dynamic CKDS update services, unless they are modified by user-written exits, check for uniqueness according to these rules prior to making any change to the CKDS.

Syntax of the ADD and UPDATE Control Statements

The ADD and UPDATE control statements use the same keywords. The ADD control statement adds new keys to the CKDS. UPDATE changes existing key entries. Use the ADD or UPDATE control statement to specify that KGUP generate a key value or import a key value that you provide.

Refer to Figure 10 on page 147 for the syntax of the ADD and UPDATE control statements.

```
{ADD | UPDATE}

  {LABEL(label1[,...,label64]) | RANGE(start-label,end-label)}

  TYPE(key-type)

  [ALGORITHM(DES|AES)]

  [OUTTYPE(key-type)]

  [TRANSKEY(key-label1[,key-label2]) | CLEAR]

  [NOCV]

  [LENGTH(n)]

  [SINGLE | DOUBLEO]

  [KEY(key-value1[,...,key-value4])]

  [KEYUSAGE(key-usage-value1[,...,key-usage-value2])]

  [DKYGENKYUSAGE(key-usage-value1[,...,key-usage-value2])]
```

Figure 10. ADD and UPDATE Control Statement Syntax

LABEL (label1[,...,label64])
This keyword defines the names of the key entries for KGUP to process within the CKDS. KGUP processes a separate entry for each label. If you specify more than one label on an ADD or UPDATE control statement, the program uses identical key values in each entry.

You must specify at least one key label, and you can specify up to 64 labels with the LABEL keyword. For the general rules about key label conventions and uniqueness, see "General Rules for CKDS Records" on page 146.

On a KGUP control statement, you must specify either the LABEL or RANGE keyword. When you supply a key value on the control statement with the KEY keyword, you must specify the LABEL keyword.

RANGE (start-label, end-label)
This keyword defines the range of the multiple labels that you want KGUP to create or maintain within the CKDS.

The label consists of between 2 and 64 characters that are divided as follows:

- The first 1 to 63 characters are the label base. These characters must be identical on both the start-label and end-label and are repeated for each label in the range. For the general rules about key label conventions and uniqueness, see "General Rules for CKDS Records" on page 146.
- The last 1 to 4 characters form the suffix. The number of digits in the start-label and end-label must be the same, and the characters must all be numeric. These numeric characters establish the range of labels KGUP creates. The start-label numeric value must be less than the end-label numeric value.

KGUP creates a separate CKDS entry for each label including the start and end labels. The program generates a different key value for each entry it creates.

You cannot use the RANGE keyword when you supply a key value to KGUP. Only use RANGE to generate a key value. The RANGE and KEY keywords are mutually exclusive.

On a KGUP control statement, you must specify either the LABEL or RANGE keyword.

TYPE (key-type)

This keyword specifies the type of key you want KGUP to process. You can specify only one key type for each control statement. For EXPORTER, IMPORTER, IPINENC, PINGEN, PINVER, and OPINENC key types, KGUP allows keys with the same labels but different key types. You can specify any of the key types in the following table.

Table 69. Key types

Key Type	Algorithm	Usage	Notes
CIPHER	AES	Data-encrypting key for the CSNBSAD and CSNBSAE services	128-, 192, or 256-bit key
CIPHER	DES	Data-encrypting key for the CSNBDEC and CSNBENC services	Single- or double-length key
CIPHERXI	DES	Input cipher translate key for CSNCTT2 and CSNCTT3 services	Double-length key May not have replicated key values. The September, 2012 or later Licensed Internal Code is required.
CIPHERXL	DES	Input cipher translate key for CSNCTT2 and CSNCTT3 services	Double-length key May not have replicated key values. The September, 2012 or later Licensed Internal Code is required.
CIPHERXO	DES	Output cipher translate key for CSNCTT2 and CSNCTT3 services	Double-length key May not have replicated key values. The September, 2012 or later Licensed Internal Code is required.
CLRAES		Clear AES data-encrypting key for the CSNBSYD and CSNBSYE services	128-, 192, or 256-bit key
CLRDES		Clear DES data-encrypting key for the CSNBSYD and CSNBSYE services	Single-, double-, or triple-length key
DATA	AES, DES	Data-encrypting key for the CSNBDEC, CSNBENC, CSNBSAD, and CSNBSAE services	Single-, double-, or triple-length key for DES 128-, 192-, or 256-bit key for AES
DATAM	DES	Double-length MAC generation key	Double-length key DOUBLEO not allowed
DATAMV	DES	Double-length MAC verification key	Double-length key DOUBLEO not allowed
DECIPHER	DES	Data-decrypting key for the CSNBDEC service	Single- or double-length key.

Table 69. Key types (continued)

Key Type	Algorithm	Usage	Notes
DKYGENKY*	AES, DES	Diversified key generating key for CSNBDKG and CSNBDKG2 services	Double-length key for DES 128-, 192-, or 256-bit key for AES
ENCIPHER	DES	Data-encrypting key for the CSNBENC service	Single- or double-length key
EXPORTER	AES, DES	Exporter key-encrypting key	Double-length key for DES 128-, 192, or 256-bit key for AES
IMPORTER	AES, DES	Importer key-encrypting key	Double-length key for DES 128-, 192, or 256-bit key for AES
IMPPKA	DES	Limited authority importer key-encrypting key	Double-length key
IPINENC	DES	Input PIN encryption key	Double-length key
KEYGENKY*	DES	Key generating key for DUKPT. Used with CSNBPTR, CSNBPTV, CSNBDKG, and CSNBUKD services	Double-length key
MAC*	AES	MAC generation and verification key	128-, 192-, or 256-bit key for AES
MAC	DES	MAC generation key	Single- or double-length key
MACVER	DES	MAC verification key	Single- or double-length key
NULL	AES, DES	Used to create a null CKDS entry	
OPINENC	DES	Output PIN encryption key	Double-length key
PINCALC*	AES	PIN calculation key	128-, 192-, or 256-bit key
PINGEN	DES	PIN generating key	Double-length key
PINPROT*	AES	PIN protection key	128-, 192-, or 256-bit key
PINPRW*	AES	PIN reference value key	128-, 192-, or 256-bit key
PINVER	DES	PIN verification key	Double-length key

All these types of keys are stored in the CKDS.

Note:

1. For compatibility with previous releases of CSF, KGUP stores internal versions of DATAM and DATAMV keys in the CKDS under the key types of MACD and MACVER, respectively.
2. Key types CIPHERXI, CIPHERXL, and CIPHERXO have control vectors with guaranteed unique key halves. Key-encrypting keys used to wrap these key types must have control vectors with guaranteed unique key halves. These key-encrypting keys can be generated using KGUP by specifying the DOULBEO keyword in the control statement.
3. The key types marked with an asterisk (*) require additional information to create the key. See the KEYUSAGE keyword for the values that must be specified.

ALGORITHM(DES|AES)

This keyword defines the algorithm of the key you are generating. DES is the default value except for key types not supported for the DES algorithm. When only one algorithm is supported for the key type, the keyword is optional. The supported algorithms for all key types is listed in the table under the TYPE keyword. Generated operational keys will be encrypted under the respective master key.

Note:

- To use an algorithm, the master key of the algorithm must be active.
- If you are going to create AES keys that use the variable-length format key token, the CKDS must be a variable-length record format CKDS and the key output data set must have a longer LRECL.

OUTTYPE (key-type)

This keyword specifies the type of complementary key you want KGUP to generate for export. This keyword is valid only when you are requesting KGUP to generate keys and you also specify the CLEAR or TRANSKEY keywords.

OUTTYPE is mutually exclusive with the KEY keyword.

Refer to Table 70 for a list of the default and optional complementary key types for each of the 11 different key types. If OUTTYPE is not specified, KGUP generates the default complementary key that is shown in this table.

Table 70. Default and Optional OUTTYPES Allowed for Each Key TYPE

Type	Algorithm	OUTTYPE (Default)	OUTTYPE (Allowed)
CIPHER	AES	CIPHER	CIPHER
CIPHER	DES	CIPHER	CIPHER, CIPHERXI, CIPHERXL, CIPHERXO, ENCIPHER, DECIPHER
CIPHERXI	DES	CIPHERXO	CIPHER, CIPHERXO, ENCIPHER
CIPHERXL	DES	CIPHERXL	CIPHER, CIPHERXL
CIPHERXO	DES	CIPHERXI	CIPHER, CIPHERXI, DECIPHER
CLRAES		Not Allowed	Not Allowed
CLRDES		Not Allowed	Not Allowed
DATA	AES	Not Allowed	Not Allowed
DATA	DES	DATA	DATA
DATAM	DES	DATAMV	DATAM, DATAMV
DATAMV	DES	Not Allowed	Not Allowed
DECIPHER	DES	ENCIPHER	CIPHER, CIPHERXO, ENCIPHER
DKYGENKY*	AES, DES	DKYGENKY*	DKYGENKY*
ENCIPHER	DES	DECIPHER	CIPHER, CIPHERXI, DECIPHER
EXPORTER	AES, DES	IMPORTER	IMPORTER
IMPORTER	AES, DES	EXPORTER	EXPORTER
IMPPKA	DES	EXPORTER	EXPORTER

Table 70. Default and Optional OUTTYPES Allowed for Each Key TYPE (continued)

Type	Algorithm	OUTTYPE (Default)	OUTTYPE (Allowed)
IPINENC	DES	OPINENC	OPINENC
KEYGENKY*	DES	KEYGENKY*	KEYGENKY*
MAC*	AES	MAC*	MAC*
MAC	DES	MACVER	MAC, MACVER
MACVER	DES	Not Allowed	Not Allowed
NULL	AES, DES	Not Allowed	Not Allowed
OPINENC	DES	IPINENC	IPINENC
PINCALC*	AES	Not Allowed	Not Allowed
PINGEN	DES	PINVER	PINVER
PINPROT*	AES	PINPROT*	PINPROT*, CIPHER
PINPRW*	AES	PINPRW*	PINPRW*
PINVER	DES	Not Allowed	Not Allowed

Note: The key types marked with an asterisk (*) require additional information to create the key and the key's complement. See the KEYUSAGE keyword for the values that must be specified.

TRANSKEY (key-label1[,key-label2])

This keyword identifies the label of a transport key that already exists in the CKDS. KGUP uses the transport key either to decrypt an imported key value or to encrypt a key value to send to another system. The algorithm of the transport key must match the key being wrapped, that is, an AES key must be wrapped with an AES transport key.

When KGUP generates a key, the program enciphers the key under the appropriate master key. KGUP may also generate a key value that can be used to create the key's complement. You can have KGUP encrypt the key value with a transport key. On the control statement, use the TRANSKEY keyword to specify an EXPORTER key-encrypting key that KGUP should use to encipher the complementary key. You can send the encrypted key value to another system to create the complementary key.

When you generate an importer key-encrypting key to encipher a key stored with data in a file, you can request that KGUP not generate the complementary export key-encrypting key. You do this by not specifying the TRANSKEY or CLEAR keyword. This is also true for CIPHER, DATA, and MAC keys.

For DES key types: When you input a key value that is in importable form, the key that is specified by the KEY keyword is enciphered under an IMPORTER key-encrypting key. KGUP reenciphers the key value from under the transport key to under a master key variant. On the control statement, you use the TRANSKEY keyword to specify the transport key that enciphers the key.

You can import or export a new version of a key that is encrypted under the current version of the same key. You can do this by specifying the same key label in the TRANSKEY keyword as in the LABEL or RANGE keyword on an UPDATE control statement.

Your site can generate keys for key exchange between two other sites. These sites do not need to know the clear value of the keys used for this

communication. KGUP generates control statements that you send to the sites. Then the sites' KGUPs establish the keys they need for key exchange.

To do this procedure, submit an ADD or UPDATE control statement with two TRANSKEY key labels. The first TRANSKEY label identifies the transport key that is valid between your site and the first recipient site. The second TRANSKEY label identifies the transport key that is valid between your site and the second recipient site. KGUP generates of a pair of control statements to create the complementary pair of keys that are needed at the two sites.

Note: You cannot specify two DES NOCV key-encrypting keys. For more information about control vectors, see the description of the NOCV keyword.

The TRANSKEY keyword and the CLEAR keyword are mutually exclusive.

If you have specified a key type of NULL, CLRDES or CLRAES for the TYPE keyword, you cannot use the TRANSKEY keyword.

CLEAR

This keyword indicates that either:

- You are supplying an unencrypted key value with the KEY keyword.
- KGUP should create a control statement that generates an unencrypted complementary key value.

You can supply either encrypted or unencrypted key values to KGUP with the KEY keyword. On the control statement to supply the unencrypted key, you specify the CLEAR keyword.

When KGUP generates a key, KGUP enciphers the key under a master key variant. KGUP may also generate a key value to be used to create the key's complement. KGUP can create the complementary key value in unencrypted form. To generate an unencrypted complementary key value, you specify the CLEAR keyword. Your ICSF system must be in special secure mode to use this keyword.

The CLEAR keyword and the TRANSKEY keyword are mutually exclusive. You cannot use the CLEAR keyword on a control statement when you use the TRANSKEY keyword. You cannot use the CLEAR keyword if you specify a NULL, CLRDES or CLRAES key for the TYPE keyword.

NOCV

To exchange keys with systems that do not recognize CCA key tokens, ICSF provides a way to by-pass transport key variant processing. KGUP or an application program encrypts a key under the transport key itself not under the transport key variant. This is called NOCV processing.

The NOCV keyword indicates that the key that is generated or imported is a DES transport key to use in NOCV processing. The transport key has the NOCV flag set in the key control information when stored in the CKDS.

Note: To create keys for NOCV processing, NOCV-Enablement keys must exist. For a description of how to create NOCV-Enablement keys, see 'Initializing the CKDS and PKDS at First-Time Startup'.

The NOCV keyword is only valid for generating transport keys. The keyword is not valid if you specify the TRANSKEY keyword with two transport key labels.

LENGTH(n), SINGLE and DOUBLEO

The LENGTH keyword specifies the length of the key value. Specifying the length of the key is optional. If the length is not specified, the default length will be used.

For AES keys and CLRAES, LENGTH(16) generates a 128-bit key, LENGTH(24) generates a 192-bit key, and LENGTH(32) generates a 256-bit key. The SINGLE and DOUBLEO keywords are not allowed.

For CLRDES keys, LENGTH(8) generates a single-length key, LENGTH(16) generates a double-length key and LENGTH(24) generates a triple-length key. The SINGLE and DOUBLEO keywords are not allowed.

For DES keys:

- LENGTH(8) generates a single-length key, LENGTH(16) generates a double-length key, and LENGTH(24) generates a triple-length key (DATA only).
- For most double-length key types, LENGTH(8) or SINGLE in an ADD or UPDATE statement causes KGUP to generate a double-length key with both key halves the same. On the KGUP panel, you can achieve this by specifying 8 in the LENGTH field for a double-length key type.
- For most double-length key types, specifying DOUBLEO causes KGUP to create a double length key with guaranteed unique key halves. The control vector is modified to indicate this. A key with this control vector can't be used on systems with the Cryptographic Coprocessor Feature.

In any case, LENGTH is used only for generating keys. If you are specifying clear or encrypted key parts, do not use the LENGTH keyword (and do not fill in a value for LENGTH on the KGUP panel).

- The LENGTH keyword and the KEY keyword are mutually exclusive.
- The SINGLE and DOUBLEO keywords are mutually exclusive.
- The SINGLE and KEY keywords are mutually exclusive.
- The DOUBLEO keyword can be specified with the KEY keyword when two different key values are supplied. The control vector will be modified.

KEY (key-value[,key-value[,key_value[,key_value]]])

This keyword allows you to supply KGUP with a key value. KGUP can use this key value to add a key or update a key entry.

If you do not specify this keyword, KGUP generates the key value for you. You cannot use the RANGE keyword or the LENGTH keyword with this keyword. Each key part consists of exactly 16 characters that represent 8 hexadecimal values.

CAUTION:
KGUP does not create complementary key control statement for existing key labels, nor new a key label that has CLEAR parm specified in the KGUP statement.

This keyword is required when you specify either DATAMV, MACVER, or PINVER for the TYPE keyword. Because these type of keys require a complementary key to be used, you must always supply values for these types of keys.

This keyword is required when you specify CIPHERXI, CIPHERXO, DECIPHER, or ENCIPHER for the TYPE keyword and the TRANSKEY and

OUTTYPE are not supplied. Because these type of keys require a complementary key to be used, you must always supply values for these types of keys.

For a double-length key, supply two key values. If you supply only one key value, KGUP will duplicate the key value as the second key value. KGUP concatenates these two identical values, and then stores and uses the key as if the key was double-length. For key types CIPHERXI, CIPHERXL, and CIPHERXO, the two keys values must be supplied and can not be the same value.

For double-length keys, when you use the TRANSKEY keyword with the KEY keyword, the transport key you specify is the importer key that encrypts the key value. If you supply only one key value for a double-length key and also specify TRANSKEY, the TRANSKEY must be an NOCV importer.

For MAC and MACVER types, you can supply one or two key values.

For a DES DATA or CLRDES key, you can supply the key in one, two, or three parts.

For an AES DATA or CLRAES key, you must supply two, three or four parts.

For an AES CIPHER, EXPORTER or IMPORTER key, you must supply two, three or four parts. Note that when the TRANSKEY keyword is specified with these keys, KGUP will not create an entry in the Control Statement Output data set.

KEYUSAGE(key-usage-value1[,...,key-usage-value2])

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

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

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

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

```
KEYUSAGE( 'CVVKEY-A' )
```

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

Table 71. Usage values for key types

Key type	Key algorithm	Key Usage Values
CIPHER	AES	The following values are optional: C-XLATE, V1PYLD and One or both may be specified: DECRYPT, ENCRYPT. **Note:** The key generated when KEYUSAGE is not specified will have only the DECRYPT and ENCRYPT key-usage. This is the default.

Table 71. Usage values for key types (continued)

Key type	Key algorithm	Key Usage Values
DKYGENKY	DES	One of the following must be specified: DKYL0, DKYL1, DKYL2, DKYL3, DKYL4, DKYL5, DKYL6, DKYL7 and One of the following must be specified: DALL, DDATA, DEXP, DIMP, DMAC, DMKEY, DMPIN, DMV, DPVR
DKYGENKY	AES	One of the following must be specified: D-PPROT, D-PCALC, D-PPRW and The following values are required: DKYL0, KUF-MBE, DKYUSAGE
DKYGENKY	AES	The following values are required: D-MAC, DKYL0, DKYUSAGE and One of the following value must be specified: KUF-MBE, KUF-MBP
DKYGENKY	AES	One of the following must be specified: D-ALL, D-CIPHER, D-EXP, D-IMP and The following value is required: DKYL0
EXPORTER	AES	The following value is optional: V1PYLD
IMPORTER	AES	The following value is optional: V1PYLD
KEYGENKY	DES	One of the following must be specified: UKPT, CLR8-ENC
MAC	DES	One of the following may be specified: ANY-MAC, CVVKEY-A, CVVKEY-B
MACVER	DES	One of the following may be specified: ANY-MAC, CVVKEY-A, CVVKEY-B
MAC	AES	One of the following must be specified: GENERATE, GENONLY, VERIFY and The following value must be specified: CMAC and One of the following is optional: DKPINOP, DKPINAD1, DKPINAD2 **Note:** • One of DKPINOP, DKPINAD1, or DKPINAD2 is required for keys to be used with the DK PIN services. • When DKPINOP, DKPINAD1, or DKPINAD2 is specified, GENERATE is not allowed.
PINCALC	AES	Three values must be specified: GENONLY, DKPINOP, and CBC.

Table 71. Usage values for key types (continued)

Key type	Key algorithm	Key Usage Values
PINPROT	AES	One of the following must be specified: ENCRYPT, DECRYPT and One of the following must be specified: DKPINOPP, DKPINOP, DKPINAD1 and The following value must be specified: CBC
PINPRW	AES	One of the following must be specified: GENONLY, VERIFY and The following values must be specified: DKPINOP, CMAC

Note:

- **DES Diversified Key Generating Keys:** The subtype field specifies the hierarchical level of the DKYGENKY. If the subtype is non-zero, the DKYGENKY can only generate another DKYGENKY key with the hierarchy level decremented by one. If the subtype is zero, the DKYGENKY can only generate the final diversified key (a non-DKYGENKY key) with the key type specified by the usage bits.
- **PINPROT Keys:** When specifying an AES CIPHER as the OUTTYPE for an AES PINPROT key, the key usage values must be ENCRYPT and DKINOPP. The key usage value for the AES CIIPHER key is DECRYPT.

Table 72. Meaning of usage values

Key Usage Value	Key types	Meaning
ANY-MAC	MAC, MACVER	The MAC usage field (control vector offset 0-3) is set to '0000'b. There is no restriction for this key. This is the default value.
C-XLATE	CIPHER	Restricts the key to be used with the cipher text translate2 service only.
CBC	PINCALC, PINPRW	Use the CBC encryption mode.
CLR8-ENC	KEYGENKY	The CLR8-ENC key usage bit (control vector offset 19) is set to '1'b. The key may only be used with the 'CLR8-ENC' rule array keyword for CSNBDKG.
CMAC	MAC, PINPROT	Use the CMAC algorithm.
CVVKEY-A	MAC, MACVER	The MAC usage field (control vector offset 0-3) is set to '0010'b. When this key is used with CSNBCVG or CSNBCVV, it can only be used as the key A parameter. This is is valid with single- and double-length keys.
CVVKEY-B	MAC, MACVER	The MAC usage field (control vector offset 0-3) is set to '0011'b. When this key is used with CSNBCVG or CSNBCVV, it can only be used as the key B parameter. This is valid with single-length keys.

Table 72. Meaning of usage values (continued)

Key Usage Value	Key types	Meaning
D-ALL	DKYGENKY	All key types may be derived except DKYGENKY keys.
D-CIPHER	DKYGENKY	CIPHER keys may be derived.
D-EXP	DKYGENKY	EXPORTER keys may be derived.
D-IMP	DKYGENKY	IMPORTER keys may be derived.
D-MAC	DKYGENKY	MAC keys may be derived.
D-PCALC	DKYGENKY	PINCALC keys may be derived.
D-PPROT	DKYGENKY	PINPROT keys may be derived.
D-PPRW	DKYGENKY	PINPRW keys may be derived.
DALL	DKYGENKY	All key types may be generated except DKYGENKY and KEYGENKY keys. Usage is restricted by an access control point. See Diversified key generate callable service.
DDATA	DKYGENKY	Generate single- and double-length DATA keys
DECRYPT	PINPROT CIPHER	This key can be used to decrypt DK PIN blocks. This key can be used to decrypt data.
DEXP	DKYGENKY	Generate EXPORTER and OKEYXLAT keys
DIMP	DKYGENKY	Generate IMPORTER and IKEYXLAT keys
DKPINAD1	MAC, PINPROT	This key may be used in the DK PIN protection methods to create or verify a pin block to allow the changing of the account number associated with a PIN.
DKPINAD2	MAC	This key may be used in the DK PIN protection methods to create or verify an account change string to allow the changing of the account number associated with a PIN.
DKPINOP	MAC, PINCALC, PINPROT, PINPRW	This key may be used in the DK PIN protection methods as a general-purpose key. It may not be used as a special-purpose key.
DKPINOPP	PINPROT	This key is to be used to encrypt a PBF-1 format pin block for the specific purpose of creating a DK PIN mailer.
DKYL0	DKYGENKY	Specifies that this key-generating key can be used to derive the key specified by the Key derivation and Derived key usage controls (AES) or control vector (DES).
DKYL1	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL0.
DKYL2	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL1.
DKYL3	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL2.
DKYL4	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL3.
DKYL5	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL4.

Table 72. Meaning of usage values (continued)

Key Usage Value	Key types	Meaning
DKYL6	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL5.
DKYL7	DKYGENKY	Specifies that this key-generating key can be used to derive a DKYGENKY with a subtype of DKYL6.
DKYUSAGE	DKYGENKY	Specifies that the DKYUSAGE keyword identifies key usage information for the key to be derived by the DKYGENKY. This value is required when the key type to be derived is MAC, PINCALC, PINPROT and PINPRW. Not valid for D-ALL, D-CIPHER, D-IMP and D-EXP.
DMAC	DKYGENKY	Generate single- and double-length MAC keys
DMKEY	DKYGENKY	Generate secure messaging keys for encrypting keys
DMPIN	DKYGENKY	Generate secure messaging keys for encrypting PINs
DMV	DKYGENKY	Generate single- and double-length MACVER keys
DPVR	DKYGENKY	Generate PINVER keys
ENCRYPT	PINPROT CIPHER	This key can be used to encrypt DK PIN blocks. This key can be used to encrypt data.
GENERATE	MAC	This key can generate and verify MACs.
GENONLY	MAC, PINCALC, PINPRW	This key can be used to only generate data (MACs, PINs, or PRWs).
KUF-MBE	DKYGENKY	Specifies that the key usage fields of the key to be generated must be equal to the related generated key usage fields of the DKYGENKY generating key. Not valid for D-ALL, D-CIPHER, D-IMP and D-EXP.
KUF-MBP	DKYGENKY	Specifies that the key usage fields of the key to be generated must be permitted based on the related generated key usage fields of the DKYGENKY generating key. The key to be derived is not permitted to have a higher level of usage than the related key usage fields permit. The key to be derived is only permitted to have key usage that is less than or equal to the related key usage fields. Not valid for D-ALL, D-CIPHER, D-IMP and D-EXP.
TRANSLAT	CIPHER	Restricts the key to be used with the cipher text translate2 service only.
UKPT	KEYGENKY	The UKPT key usage bit (control vector offset 18) is set to '1'b. The key may only be used in the CSNBPTR and CSNBPVR services.
VERIFY	MAC, PINPRW	This key can be used to verify data (MACs or PRWs).
V1PYLD	CIPHER, EXPORTER, IMPORTER	The generated key or keys will have version 1 (fixed-length) format of the payload for the variable-length symmetric key token. Applies to AES keys only.

Note:

- **Diversified Key Generating Key Note:** The subtype field specifies the hierarchical level of the DKYGENKY. If the subtype is non-zero, then the DKYGENKY can only generate another DKYGENKY key with the hierarchy level decremented by one. If the subtype is zero, the DKYGENKY can only generate the final diversified key (a non-DKYGENKY key) with the key type specified by the usage bits.
- **PINPROT Keys:** When specifying an AES CIPHER as the OUTTYPE for an AES PINPROT key, the key usage values must be ENCRYPT and DKINOPP. The key usage value for the AES CIIPHER key is DECRYPT.
- **AES MAC Keys:** When DKPINOP, DKPINAD1, or DKPINAD2 is specified, GENERATE is not allowed.

Complementary key-usage values

When a pair of keys is generated, one for the local system and the other for a remote system,

- **For the AES CIPHER key type**, the key usage for the complementary key is determined from the values from the KEYUSAGE keyword as shown in Table 73. The other values do not have a complementary value and are copied.

Table 73. Complementary key-usage values for AES CIPHER

Key usage values	Complementary key usage values
ENCRYPT, DECRYPT	ENCRYPT, DECRYPT
ENCRYPT	DECRYPT
DECRYPT	ENCRYPT

- **For the AES MAC key type**, the key usage for the complementary key is determined from the values from the KEYUSAGE keyword as shown in Table 74. The other values do not have a complementary value and are copied. Note that for any key generated for the DK PIN methods, the local system gets the GENONLY key-usage. VERIFY key-usage is not allowed.

Table 74. Complementary key-usage values for AES MAC

Key usage values	Complementary key usage values
GENERATE	GENERATE
GENONLY	VERIFY
GENONLY, DKPINOP	VERIFY, DKPINOP
GENONLY, DKPINAD1	VERIFY, DKPINAD1
GENONLY, DKPINAD2	VERIFY, DKPINAD2
VERIFY	GENONLY

- **For the AES PINPROT key type**:
  - When TRANSKEY is specified, the ENCRYPT value is allowed for the local system and DECRYPT values is allowed for the remote system.
  - When CLEAR is specified, ENCRYPT and DECRYPT are complementary values.
  - The other values do not have a complementary value and are copied.
- **For the AES PINPRW key types**:
  - When TRANSKEY is specified, the GENONLY value is allowed for the local system and VERIFY values is allowed for the remote system.

- When CLEAR is specified, GENONLY and VERIFY are complementary values.
- The other values do not have a complementary value and are copied.

- **For the AES DKYGENKY key type**, the key usage values for the complementary key are the complement of the generated key. There are restrictions for the values specified in the DKYGENKEYUSAGE keyword. See the DKYGENKEYUSAGE keyword description.
- **For all other key types**, both keys are generated with the same key-usage values.

DES
This keyword is no longer supported but is tolerated.

DKYGENKYUSAGE(key-usage-value1[,...,key-usage-value2])
This keyword defines key usage values to be supplied for the AES DKYGENKY key being generated. This keyword is required when the DKYUSAGE value is specified in the KEYUSAGE keyword.

The following values have been defined. The usage values are specific to the key type to be derived. The values can only be specified for the key type indicated in Table 75 and Table 76 on page 161. The values for the specific key types are detailed in this document in the Key Token Build2 callable service description.

Note: Any value with a non-alphanumeric character must be enclosed in quotes when specified with the DKYGENKYUSAGE keyword. For example: DKYGENKYUSAGE('CVVKEY-A').

Table 75. Values by type for DKYGENKYUSAGE

Type of key to be derived	DKYGENKYUSGE values
MAC	One of the following values is required: GENERATE, GENONLY, VERIFY and The following value is required: CMAC and One of the following values is optional: DKPINAD1, DKPINAD2, DKPINOP **Note:** • One of DKPINOP, DKPINAD1, or DKPINAD2 is required for keys to be used with the DK PIN services. • When DKPINOP, DKPINAD1, or DKPINAD2 is specified, GENERATE is not allowed.
PINCALC	The following values are required: GENONLY, CBC, DKPINOP.

Table 75. Values by type for DKYGENKYUSAGE (continued)

Type of key to be derived	DKYGENKYUSGE values
PINPROT	One of the following values is required: DECRYPT, ENCRYPT and The following value is required: CBC and One of the following values is required: DKPINAD1, DKPINOP, DKPINOPP
PINPRW	One of the following values is required: GENONLY, VERIFY and The following values are required: CMAC, DKPINOP

Table 76. Meaning of usage values

Value	Key types	Description
CBC	PINPROT, PINCALC	The derived key must use the CBC encryption mode.
CMAC	MAC, PINPRW	The derived key must use the CMAC algorithm.
DECRYPT	PINPROT	The derived key may be used to decrypt PIN blocks.
DKPINAD1	MAC, PINPROT	The derived key may be used to create or verify a pin block to allow changing the account number associate with a PIN for the DK PIN methods.
DKPINAD2	MAC	The derived key may be used to create or verify an account change string to allow changing the account number associated with a PIN for the DK PIN methods.
DKPINOP	MAC, PINCALC, PINPROT, PINPRW	The derived key may be used as a general purpose key for the DK PIN methods.
DKPINOPP	PINPROT	The derived key may be used to encrypt a PIN block for the specific purpose of creating a PIN mailer for the DK PIN methods.
ENCRYPT	PINPROT	The derived key may be used to encrypt PIN blocks.
GENERATE	MAC	The derived key may be used to generate and verify MACs.
GENONLY	MAC, PINCALC	The derived key may be used to generate MACs or PINs.
VERIFY	MAC	The derived key may be used to verify MACs.

Complementary DKYGENKY usage values

When a pair of DKYGENKY keys is generated, one for the local system and the other for a remote system, the complementary key will have a different value as shown in Table 77. Values that do not appear in the table are copied for the complementary key.

Table 77. Complementary values for usage values

Type of key to be derived	DKYGENKY usage value	Complementary value
MAC	GENERATE	GENERATE
MAC	GENONLY	VERIFY
MAC	VERIFY	GENONLY
MAC with DKPINOP, DKPINAD1 or DKPINAD2	GENONLY	VERIFY
PINCALC	Not allowed	Not allowed
PINPROT	ENCRYPT	DECRYPT
PINPRW	GENONLY	VERIFY

Attention: NOCV processing takes place automatically when KGUP or an application specifies the use of a transport key that was generated by KGUP with a NOCV keyword specified.

The use of NOCV processing eliminates the ability of the system that generates the key to determine the use of the key on a receiving system. Therefore, access to these keys should be strictly controlled. For a description of security considerations, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

Callable services affected by key store policy

This topic provides the resource names for the new callable services that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods.

Only the names of the 31-bit versions of the callable services are listed. However, 64-bit versions of the callable services and the ALET qualified versions of the services are also covered by the key store policy. The callable services that are affected by the TOKEN_CHECK key store policy controls are in the table below.

Table 78. Callable services and parameters affected by key store policy

ICSF callable service	31-bit name	Parameter checked
Diversified Key Generate2	CSNBDKG2	generating_key_identifier
DK PAN Modify in Transaction	CSNBDPMT	CMAC_FUS_key_identifier IPIN_encryption_key_identifier PRW_key_identifier new_PRW_key_identifier
DK PIN Change	CSNBDPC	PRW_MAC_key_identifier cur_IPIN_encryption_key_identifier new_IPIN_encryption_key_identifier script_key_identifier script_MAC_key_identifier new_PRW_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier

Table 78. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
DK PIN Verify	CSNBDPV	PRW_MAC_key_identifier IPIN_encryption_key_identifier
DK Random PIN Generate	CSNBDRPG	PRW_MAC_key_identifier PIN_print_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier

Chapter 4. Update of z/OS Cryptographic Services ICSF System Programmer's Guide, SC14-7507-00, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF System Programmer's Guide*, SC14-7507-00, for the DK AES PIN support provided by this APAR. Refer to this source document if background information is needed.

Installation, Initialization, and Customization

This topic provides the updates that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods.

Parameters in the installation options data set

These are the exit identifiers for the new services.

Table 79. Exit Identifiers and Exit Invocations

Exit Identifiers	Exit Invocations
CSFDKG2	Gets control during the Diversified Key Generate2 callable service.
CSFDPC	Gets control during the DK PIN Change callable service.
CSFDPMT	Gets control during the DK PAN Modify in Transaction callable service.
CSFDPV	Gets control during the DK PIN Verify callable service.
CSFDRPG	Gets control during the DK Random PIN Generate callable service.

Migration

This topic provides the updates that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods.

Migrating from earlier software releases

This topic provides the updates that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods.

Callable Services

The following table summarizes the new and changed callable services for ICSF FMID HCR77A1. For complete reference information on these callable services, refer to *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Table 80. Summary of new and changed ICSF callable services

Callable service	Release	Description
Authentication Parameter Generate	HCR77A1	**New:** Generate an authentication parameter (AP) and return it encrypted under a supplied encrypting key.
ICSF Query Facility 2	HCR77A1	**New:** Provides information on the cryptographic environment as currently known by ICSF.
Recover PIN From Offset	HCR77A1	**New:** Calculate an encrypted customer-entered PIN from a PIN generating key, account information, and an offset, returnining the PIN properly formatted and encrypted under a PIN encryption key.

Table 80. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
Symmetric Key Export with Data	HCR77A1	**New:** Export a symmetric key encrypted using an RSA key, inserted in a PKCS#1 block type 2, with some extra data supplied by the application.
Cipher Text Translate2 and Cipher Text Translate2 with alet	HCR77A0	**New:** Translates the user-supplied ciphertext from one key to another key.
Control Vector Generate	HCR77A0	**Changed:** • Support CIPHERXI, CIPHERXL and CIPHERXO key types. • Support DOUBLE-O rule_array keyword.
Diversified Key Generate2	HCR77A0	**New:** Derive keys using a key-generating key.
DK PIN Change	HCR77A0	**New:** Allow a customer to select a personal PIN.
DK PIN Verify	HCR77A0	**New:** Verify an ISO-1 PIN.
DK PAN Modify in Transaction	HCR77A0	**New:** This service is used to obtain a new PIN reference value (PRW) for an existing PIN when the account information has changed.
DK Random PIN Generate	HCR77A0	**New:** Generate a random PIN and PIN reference value.
ECC Diffie-Hellman	HCR77A0	**Changed:** • Support CIPHERXI, CIPHERXL and CIPHERXO key types. • Support creation of DES keys with guaranteed unique key halves.
Key Export	HCR77A0	**Changed:** Support CIPHERXI, CIPHERXL and CIPHERXO key types.
Key Generate	HCR77A0	**Changed:** • Support CIPHERXI, CIPHERXL and CIPHERXO key types. • Support DOUBLE-O key_length.
Key Generate2	HCR77A0	**Changed:** • Support generating AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW keys for DK AES PIN services. • Support generating AES CIPHER keys for use in Cipher Text Translate2 callable service.
Key Import	HCR77A0	**Changed:** Support CIPHERXI, CIPHERXL and CIPHERXO key types.
Key Part Import2	HCR77A0	**Changed:** Support AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW keys for DK AES PIN services.
Key Test2	HCR77A0	**Changed:** Support AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW keys for DK AES PIN services.
Key Token Build	HCR77A0	**Changed:** • Support CIPHERXI, CIPHERXL and CIPHERXO key types. • Support DOUBLE-O rule_array keyword.
Key Token Build2	HCR77A0	**Changed:** • Support generating AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW key tokens. • Support C-XLATE keyword for AES CIPHER key type.

Table 80. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
Key Translate2	HCR77A0	**Changed:** Support changing variable-length key tokens with variable-length payloads to fixed-length payloads.
ICSF Query Facility	HCR77A0	**Changed:** Retrieve weak PIN table from coprocessor.
Multiple Secure Key Import	HCR77A0	**Changed:** Support CIPHERXI, CIPHERXL and CIPHERXO key types
PKA Key Generate	HCR77A0	**Changed:** Support generating RSA keys that can be wrapped by AES keys.
PKA Key Import	HCR77A0	**Changed:** Support importing RSA keys that are wrapped by an AES key-encrypting key.
PKA Key Token Build	HCR77A0	**Changed:** Support building RSA-AESC and RSA-AESM skeleton tokens.
PKA Key Token Change	HCR77A0	**Changed:** Support reenciphering RSA keys wrapped by an ECC master key.
PKA Key Translate	HCR77A0	**Changed:** Support translating the object protection key (OPK) in a RSA private key token from a DES key to an AES key.
Restrict Key Attribute	HCR77A0	**Changed:** • Support AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW keys for DK AES PIN services. • Support C-XLATE rule_array keyword for AES CIPHER keys. • Support DOUBLE-O rule_array keyword for DES keys.
Secure Key Import	HCR77A0	**Changed:** Support CIPHERXI, CIPHERXL and CIPHERXO key types.
Secure Key Import2	HCR77A0	**Changed:** Support AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW keys for DK AES PIN services.
Symmetric Key Export	HCR77A0	**Changed:** Support AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW keys for DK AES PIN services.
Symmetric Key Import2	HCR77A0	**Changed:** Support AES DKYGENKY, MAC, PINCALC, PINPROT, and PINPRW keys for DK AES PIN services.
Unique Key Derive	HCR77A0	**New:** Use the Unique Key Derive callable service to derive a key using the Base Derivation Key and the Derivation Data. The following key types can be derived: • CIPHER • ENCIPHER • DECIPHER • MAC • MACVER • IPINENC • OPINENC • DATA token containing a PIN Key
Clear PIN Generate	HCR7790	**Changed:** Increased X9.8 PIN block security, stored PIN decimalization tables support.
Clear PIN Generate Alternate	HCR7790	**Changed:** Increased X9.8 PIN block security, stored PIN decimalization tables support.
Control Vector Generate	HCR7790	**Changed:** ANSI TR-31 key block support.
Coordinated KDS Administration	HCR7790	**New:** Support for a coordinated CKDS refresh or a coordinated CKDS reencipher and master key change.

Table 80. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
CVV Key Combine	HCR7790	**New:** Double-length CVV key support
Digital Signature Verify	HCR7790	**Changed:** 4096-bit RSA clear key hardware support.
ECC Diffie-Hellman	HCR7790	**New:** Creation of: • Symmetric key material from a pair of ECC keys using the Elliptic Curve Diffie-Hellman protocol using the Static Unified Model • "Z" - The "secret" material output from D-H process
Encrypted PIN Generate	HCR7790	**Changed:** Increased X9.8 PIN block security, stored PIN decimalization tables support.
Encrypted PIN Verify	HCR7790	**Changed:** Increased X9.8 PIN block security, stored PIN decimalization tables support.
ICSF Query Algorithm	HCR7790	**Changed:** 4096-bit RSA clear key hardware support.
ICSF Query Facility	HCR7790	**Changed:** • Increased X9.8 PIN block security, stored PIN decimalization tables support. • ECC Diffie-Hellman (ECCDH) and ECC key wrapping support. • 4096-bit RSA clear key hardware support.
Key Generate2	HCR7790	**Changed:** AES key type support
Key Part Import2	HCR7790	**Changed:** AES key type support
Key Test2	HCR7790	**Changed:** • AES key type support • ANSI TR-31 key block support.
Key Token Build	HCR7790	**Changed:** ANSI TR-31 key block support.
Key Token Build2	HCR7790	**Changed:** AES key type support
Key Translate2	HCR7790	**Changed:** AES key type support
PKA Decrypt	HCR7790	**Changed:** 4096-bit RSA clear key hardware support.
PKA Encrypt	HCR7790	**Changed:** 4096-bit RSA clear key hardware support.
PKA Key Generate	HCR7790	**Changed:** Support for External ECC Keys (ECC Keys encrypted by an AES KEK)
PKA Key Import	HCR7790	**Changed:** Support for External ECC Keys (ECC Keys encrypted by an AES KEK)
PKCS #11 Derive key	HCR7790	**Changed:** Support for hardware generated "z" value.
PKCS #11 Derive multiple keys	HCR7790	**Changed:** Support for hardware generated "z" value.
PKCS #11 Private key sign	HCR7790	**Changed:** 4096-bit RSA clear key hardware support.
PKCS #11 Public key verify	HCR7790	**Changed:** 4096-bit RSA clear key hardware support.
PKCS #11 Unwrap key	HCR7790	**Changed:** 4096-bit RSA clear key hardware support.
Restrict Key Attribute	HCR7790	**Changed:** • AES key type support • ANSI TR-31 key block support.
Secure Key Import2	HCR7790	**Changed:** AES key type support
Symmetric Algorithm Decipher	HCR7790	**Changed:** AES key type support

Table 80. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
Symmetric Algorithm Encipher	HCR7790	**Changed:** AES key type support
Symmetric Key Export	HCR7790	**Changed:** • AES key type support • Support for PKCS#1 OAEP data block formatting with the SHA-256 hash method
Symmetric Key Generate	HCR7790	**Changed:** Support for PKCS#1 OAEP data block formatting with the SHA-256 hash method
Symmetric Key Import	HCR7790	**Changed:** Support for PKCS#1 OAEP data block formatting with the SHA-256 hash method
Symmetric Key Import2	HCR7790	**Changed:** AES key type support
TR-31 Export	HCR7790	**New:** ANSI TR-31 key block support.
TR-31 Import	HCR7790	**New:** ANSI TR-31 key block support.
TR-31 Optional Data Build	HCR7790	**New:** ANSI TR-31 key block support.
TR-31 Optional Data Read	HCR7790	**New:** ANSI TR-31 key block support.
TR-31 Parse	HCR7790	**New:** ANSI TR-31 key block support.
VISA CVV Service Verify	HCR7790	**Changed:** Double-length CVV key support
VISA CVV Service Generate	HCR7790	**Changed:** Double-length CVV key support
ANSI X9.17 EDC Generate	HCR7780	**Changed:** Support for invocation in AMODE(64).
ANSI X9.17 Key Export	HCR7780	**Changed:** Support for invocation in AMODE(64).
ANSI X9.17 Key Import	HCR7780	**Changed:** Support for invocation in AMODE(64).
ANSI X9.17 Key Translate	HCR7780	**Changed:** Support for invocation in AMODE(64).
ANSI X9.17 Transport Key Partial Notarize	HCR7780	**Changed:** Support for invocation in AMODE(64).
Ciphertext Translate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Clear PIN Encrypt	HCR7780	**Changed:** Support for invocation in AMODE(64).
Clear PIN Generate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Clear PIN Generate Alternate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Control Vector Generate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Control Vector Translate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Cryptographic Variable Encipher	HCR7780	**Changed:** Support for invocation in AMODE(64).
Data Key Export	HCR7780	**Changed:** Support for invocation in AMODE(64).
Data Key Import	HCR7780	**Changed:** Support for invocation in AMODE(64).
Decipher	HCR7780	**Changed:** Support for invocation in AMODE(64).
Decode	HCR7780	**Changed:** Support for invocation in AMODE(64).
Digital Signature Generate	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
Digital Signature Verify	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
Diversified Key Generate	HCR7780	**Changed:** • Support for invocation in AMODE(64). • New rule array keywords to support enhanced key wrapping method.

Table 80. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
Encipher	HCR7780	**Changed:** Support for invocation in AMODE(64).
Encode	HCR7780	**Changed:** Support for invocation in AMODE(64).
Encrypted PIN Generate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Encrypted PIN Translate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Encrypted PIN Verify	HCR7780	**Changed:** Support for invocation in AMODE(64).
HMAC Generate	HCR7780	**New:** Support for CCA key management of HMAC keys.
HMAC Verify	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Export	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Generate2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Import	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Part Import	HCR7780	**Changed:** • Support for invocation in AMODE(64). • New rule array keywords to support enhanced key wrapping method.
Key Part Import2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Record Create	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Record Create2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Record Delete	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Record Read	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Record Read2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Record Write	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Record Write2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Test	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Test Extended	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Test2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Token Build	HCR7780	**Changed:** • Support for invocation in AMODE(64). • New rule array keywords to support enhanced key wrapping method.
Key Token Build2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Key Translate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Key Translate2	HCR7780	**New:** Support for CCA key management of HMAC keys.
MAC Generate	HCR7780	**Changed:** Support for invocation in AMODE(64).
MAC Verify	HCR7780	**Changed:** Support for invocation in AMODE(64).
MDC Generate	HCR7780	**Changed:** Support for invocation in AMODE(64).
Multiple Clear Key Import	HCR7780	**Changed:** New rule array keywords to support enhanced key wrapping method.
Multiple Secure Key Import	HCR7780	**Changed:** • Support for invocation in AMODE(64). • New rule array keywords to support enhanced key wrapping method.
One-Way Hash Generate	HCR7780	**New:** Support for invocation in AMODE(64).

Table 80. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
PIN Change/Unblock	HCR7780	**Changed:** Support for invocation in AMODE(64).
PKA Key Generate	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
PKA Key Import	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
PKA Key Token Build	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
PKA Key Token Change	HCR7780	**Changed:** • Elliptic Curve Cryptography (ECC) support. • Support for invocation in AMODE(64).
PKA Public Key Extract	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
PKDS Key Record Create	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
PKDS Key Record Delete	HCR7780	**Changed:** Elliptic Curve Cryptography (ECC) support.
PKDS Key Record Read	HCR7780	**Changed:** • Elliptic Curve Cryptography (ECC) support. • Support for invocation in AMODE(64).
PKDS Key Record Write	HCR7780	**Changed:** • Elliptic Curve Cryptography (ECC) support. • Support for invocation in AMODE(64).
Prohibit Export	HCR7780	**Changed:** Support for invocation in AMODE(64).
Prohibit Export Extended	HCR7780	**Changed:** Support for invocation in AMODE(64).
Remote Key Export	HCR7780	**Changed:** Support for invocation in AMODE(64).
Restrict Key Attribute	HCR7780	**New:** Support for CCA key management of HMAC keys.
Secure Key Import	HCR7780	**Changed:** Support for invocation in AMODE(64).
Secure Key Import2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Secure Messaging for Keys	HCR7780	**Changed:** Support for invocation in AMODE(64).
Secure Messaging for PINS	HCR7780	**Changed:** Support for invocation in AMODE(64).
SET Block Compose	HCR7780	**Changed:** Support for invocation in AMODE(64).
SET Block Decompose	HCR7780	**Changed:** Support for invocation in AMODE(64).
Symmetric Key Decipher	HCR7780	**Changed:** Additional modes of operation for protecting data.
Symmetric Key Encipher	HCR7780	**Changed:** Additional modes of operation for protecting data.
Symmetric Key Export	HCR7780	**Changed:** Support for CCA key management of HMAC keys.
Symmetric Key Generate	HCR7780	**Changed:** • Support for invocation in AMODE(64). • New rule array keywords to support enhanced key wrapping method.
Symmetric Key Import	HCR7780	**Changed:** New rule array keywords to support enhanced key wrapping method.
Symmetric Key Import2	HCR7780	**New:** Support for CCA key management of HMAC keys.
Transaction Validation	HCR7780	**Changed:** Support for invocation in AMODE(64).
Transform CDMF Key	HCR7780	**Changed:** Support for invocation in AMODE(64).
Trusted Block Create	HCR7780	**Changed:** Support for invocation in AMODE(64).
User Derived Key	HCR7780	**Changed:** Support for invocation in AMODE(64).
VISA CVV Service Generate	HCR7780	**Changed:** Support for invocation in AMODE(64).

Table 80. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
VISA CVV Service Verify	HCR7780	**Changed:** Support for invocation in AMODE(64).
PKCS #11 Derive key	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Derive multiple keys	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Generate HMAC	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Generate key pair	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Generate secret key	HCR7770	**New:** Support for PKCS #11.
PKCS #11 One-way hash generate	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Private key sign	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Pseudo-random function	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Public key verify	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Secret key decrypt	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Secret key encrypt	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Unwrap key	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Verify HMAC	HCR7770	**New:** Support for PKCS #11.
PKCS #11 Wrap key	HCR7770	**New:** Support for PKCS #11.
PKA Key Translate	HCR7770	**New:** Support for RSA private key export.
PKA Key Generate	HCR7770	**Changed:** Support for RSA private key export.
PKA Key Token Build	HCR7770	**Changed:** Support for RSA private key export.
Symmetric Key Export	HCR7770	**Changed:** Support for invocation in AMODE(64).
Symmetric Key Import	HCR7770	**Changed:** Support for invocation in AMODE(64).
Symmetric Key Encipher	HCR7770	**Changed:** Support an encrypted key in the CKDS.
Symmetric Key Decipher	HCR7770	**Changed:** Support an encrypted key in the CKDS.

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:

- HCR77A1: CSFAPG, CSFPFO, CSFSXD
- HCR77A0: CSFCTT2, CSFCTT3, CSFUDK, CSFDPV, CSFDPC, CSFDPMT, CSFDRPG, CSFDKG2
- HCR7790: CSFEDH, CSFT31X, CSFT31I, CSFCKC
- HCR7780: CSFHMG, CSFHMG1, CSFHMV, CSFHMV1, CSFKGN2, CSFKPI2, CSFKTR2, CSFKYT2, CSFRKA, CSFSKI2, CSFSYI2, CSFKRC2, CSFKRW2
- HCR7770: CSNBSYD, CSNBSYD1, CSNBSYE, CSNBSYE1, CSFPKT, CSF1DMK, CSF1DVK, CSF1SKD, CSF1SKE, CSF1HMG, CSF1HMV, CSF1OWH, CSF1PRF,

CSNBSAD, CSNBSAD1, CSNBSAE, CSNBSAE1, CSFRNGL, CSF1GKP, CSF1GSK, CSF1PKS, CSF1PKV, CSF1SAV, CSF1TRC, CSF1TRD, CSF1UWK, CSF1WPK, CSFTBC, CSFRKX

- HCR7751: CSNBSAD, CSNBSAD1, CSNBSAE, CSNBSAE1, CSFRNGL, CSF1GKP, CSF1GSK, CSF1PKS, CSF1PKV, CSF1SAV, CSF1TRC, CSF1TRD, CSF1UWK, CSF1WPK, CSFTBC, CSFRKX

Note: If no Wait List is specified, the default wait list will be used. See sample CSFWTL01 for the contents of the default wait list.

Chapter 5. Update of z/OS Cryptographic Services ICSF Messages, SC14-7509-00, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF Messages*, SC14-7509-00, for the DK AES PIN support provided by this APAR. Refer to this source document if background information is needed.

CSFGnnnn Messages (Key Generator Utility Processing)

"CSFGnnnn Messages (Key Generator Utility Processing)" describes messages that the key generator utility program (KGUP) issues. These messages are sent to the KGUP diagnostic data set (CSFDIAG). For information about defining the CSFDIAG data set, see *z/OS Cryptographic Services ICSF System Programmer's Guide*.

CSFG0224 *keyword* **SPECIFIED WITH TYPE** *keytype***.**

Explanation: There is a mismatch between *keyword* and *keytype*.

- If NOCV is specified, only key types EXPORTER or IMPORTER are allowed.
- KEYUSAGE may be specified only with key types with defined values as shown in the description for KEYUSAGE.
- DKYGENKYUSAGE may only be specified with key type DKYGENKY.
- If *keytype* CLRDES or CLRAES is specified, *keywords* CLEAR, OUTTYPE and TRANSKEY are not allowed.

System action: Processing ends for this control statement. Normal processing of the input file continues.

User response: Make sure that the keyword is valid for the key type. Rerun the key generator utility program.

CSFG0924 KEYUSAGE VALUES ARE NOT CONSISTENT.

Explanation: The KEYUSAGE values are not consistent. There may be more than one value specified where only one is allowed. Two or more values may have been specified which are not allowed to be used together. There may be one or more missing values. DKYUSAGE may not have been specified when the DKYGENKEYUSAGE keyword was specified.

System action: Processing ends for this control statement. Normal processing of the input file continues.

User response: Check the syntax of the control statement. Ensure that you specified the statement keywords and values correctly. For example, check for unpaired delimiters and missing or extraneous commas. Rerun the key generator utility program.

CSFG0994 DKYGENKYUSAGE VALUE *value* **SPECIFIED WITH** *keytype* **KEY TYPE FOR ALGORITHM** *algorithm***.**

Explanation: The DKYGENKYUSAGE *value* is not valid for the key type and algorithm specified in the control statement. The valid values are shown in the description for DKYGENKYUSAGE.

System action: Processing ends for this control statement. Normal processing of the input file continues.

User response: Check the syntax of the control statement. Ensure that you specified the statement keywords and values correctly. For example, check for unpaired delimiters and missing or extraneous commas. Rerun the key generator utility program.

CSFG1004 DKYGENKYUSAGE VALUES ARE NOT CONSISTENT.

Explanation: The DKYGENKYUSAGE values are not consistent. There may be more than one value specified where only one is allowed. Two or more values may have been specified which are not allowed to be used together. There may be one or more missing values.

System action: Processing ends for this control statement. Normal processing of the input file continues.

User response: Check the syntax of the control statement. Ensure that you specified the statement keywords and values correctly. For example, check for unpaired delimiters and missing or extraneous commas. Rerun the key generator utility program.

CSFG1014 ENCRYPTED KEY SUPPLIED FOR AES KEY.

Explanation: An encrypted key value was supplied to be imported to an AES key. This usage is not supported for AES as it is for DES.

System action: Processing ends for this control statement. Normal processing of the input file continues.

User response: Check the syntax of the control statement. Ensure that you specified the statement keywords and values correctly. Rerun the key generator utility program.

CSFG1024 KEYUSAGE VALUES REQUIRE KEY OR TRANSKEY.

Explanation: Certain KEYUSAGE values for a key type require the complementary key be generated or a key value be supplied. If, for example, an AES CIPHER key is generated with KEYUSAGE(ENCRYPT), the key can only be used for encrypting data. There is no key to decrypt the data. A key value can be supplied and a complementary key can be generated with the same key value. A complementary key can be generated and wrapped with a transport key. Complementary KEYUSAGE values are shown in the *z/OS ICSF Administrator's Guide* in the description for KEYUSGE.

System action: Processing ends for this control statement. Normal processing of the input file continues.

User response: Check the syntax of the control statement. Decide whether to provide a key value or transport key or modify the KEYUSAGE values. Rerun the key generator utility program.

Glossary

Central Credit Committee
The official English name for *Zentraler Kreditausschuss*, also known as ZKA. ZKA was founded in 1932 and was renamed in August 2011 to *Die Deutsche Kreditwirtschaft*, also known as DK. DK is an association of the German banking industry. The hybrid term in English for DK is 'German Banking Industry Committee'.

DK *Die Deutsche Kreditwirtschaft* (German Banking Industry Committee). Formerly known as ZKA.

German Banking Industry Committee
A hybrid term in English for *Die Deutsche Kreditwirtschaft*, also known as DK, an association of the German banking industry. Prior to August 2011, DK was named ZKA for *Zentraler Kreditausschuss*, or Central Credit Committee. ZKA was founded in 1932.

Printed in USA