z/OS

Cryptographic Services Integrated Cryptographic Service Facility DK AES PIN Part 2 Support - APAR OA43906

Contents

Chapter 1. Overview

This document describes changes to the Integrated Cryptographic Service Facility (ICSF) product in support of the German Banking Industry Committee (DK) PIN methods. Five new callable services are added in support of the DK PIN methods as well as two new callable services for generating and verifying MACs using AES keys:

- DK Deterministic PIN Generate (CSNBDDPG and CSNEDDPG)
- DK PAN Translate (CSNBDPT and CSNEDPT)
- DK PRW Card Number Update (CSNBDPNU and CSNEDPNU)
- DK PRW CMAC Generate (CSNBDPCG and CSNEDPCG)
- DK Regenerate PRW (CSNBDRP and CSNEDRP)
- MAC Generate2 (CSNBMGN2, CSNBMGN3, CSNEMGN2, and CSNEMGN3)
- MAC Verify2 (CSNBMVR2, CSNBMVR3, CSNEMVR2, and CSNEMVR3)

Note: All crypto coprocessors must be loaded with the same level of code. There have been several licensed internal code (LIC) released in support of the DK PIN methods. Ensure that all of the coprocessors have the same LIC level to support the function you want to use.

These changes are available through the application of the PTF for APAR OA43906 and apply to FMID HCR77A1 and HCR77A0. This document 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 Overview*, SC14-7505-00

The technical changes made to the ICSF product by the application of the PTF for APAR OA43906 are indicated in this document by a vertical line to the left of the change.

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 Part 2 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 Deterministic PIN Generate, DK PIN Change, DK PAN Modify in Transaction, DK PAN Translate, DK PRW Card Number Update, DK PRW CMAC Generate, DK Random PIN Generate, DK Regenerate PRW, MAC Generate2, MAC Verify2
PIN class These keys are used in various financial-PIN processing services.	
PINCALC	DK Deterministic PIN Generate
PINPROT	DK Deterministic PIN Generate, DK PAN Translate, DK PIN Change, DK PRW Card Number Update, DK Random PIN Generate, DK Regenerate PRW
PINPRW	DK Deterministic PIN Generate, DK PAN Modify in Transaction, DK PAN Translate, DK PIN Change, DK PIN Verify, DK PRW Card Number Update, DK Random PIN Generate, DK Regenerate PRW
Key generating class These keys are used to derive operational keys.	
DKYGENKY	Diversified Key Generate2

Managing Data Integrity and Message Authentication

To ensure the integrity of transmitted messages and stored data, ICSF provides:

- Message authentication code (MAC)
- Several hashing functions, including modification detection code (MDC), SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, RIPEMD-160 and MD5.

MAC Generation2 Callable Service (CSNBMGN2 or CSNBMGN3 and CSNEMGN2 or CSNEMGN3)

When a message is sent, an application program can generate an authentication code for it using the MAC generation2 callable service.

The callable service computes the message authentication code using FIPS-198 Keyed-Hash Message Authentication Code method for HMAC key or the CMAC (NIST SP 800-38B) algorithm for AES keys.

MAC Verification2 Callable Service (CSNBMVR2 or CSNBMVR3 and CSNEMVR2 or CSNEMVR3)

When the receiver gets the message, an application program calls the MAC verification2 callable service. The callable service verifies a MAC by generating another MAC and comparing it with the MAC received with the message. If the two codes are the same, the message sent was the same one received. A return code indicates whether the two MACs are the same.

The MAC verification callable service can use FIPS-198 Keyed-Hash Message Authentication Code method for HMAC key or the CMAC (NIST SP 800-38B) algorithm for AES keys.

DK PIN methods support

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

- "DK Deterministic PIN Generate (CSNBDDPG and CSNEDDPG)"
- "DK PAN Translate (CSNBDPT and CSNEDPT)"
- "DK PRW Card Number Update (CSNBDPNU and CSNEDPNU)"
- "DK PRW CMAC Generate (CSNBDPCG and CSNEDPCG)"
- "DK Regenerate PRW (CSNBDRP and CSNEDRP)"

DK Deterministic PIN Generate (CSNBDDPG and CSNEDDPG)

The DK Deterministic PIN Generate service is used to generate a PIN and PIN reference value (PRW) using an AES PIN calculation key. The PIN reference value is used to verify the PIN in other services.

DK PAN Translate (CSNBDPT and CSNEDPT)

The DK PAN Translate service is create an encrypted PIN block with the same PIN and a different PAN. The account data may change, but changing the PIN is to be avoided. This service creates a new encrypted PIN block and MAC on the encrypted PIN block that will be used to accept the PAN change at an authorization node.

DK PRW Card Number Update (CSNBDPNU and CSNEDPNU)

The DK PRW Card Number Update (CSNBDPNU and CSNEDPNU) service is used to generate a PIN reference value (PRW) when a replacement card is being issued. The original PAN data and PIN are used with a new card number to generate the new PRW.

DK PRW CMAC Generate (CSNBDPCG and CSNEDPCG)

The DK PRW CMAC Generate (CSNBDPCG and CSNEDPCG) service is used to generate a message authentication code (MAC) over specific values involved in an account number change transaction. The inputs include the current and new PAN and card data and the PIN reference value.

DK Regenerate PRW (CSNBDRP and CSNEDRP)

The DK Regenerate PRW (CSNBDRP and CSNEDRP) service is used to generate a new PIN reference value for a changed account number.

Verifying Data Integrity and Authenticating Messages

ICSF provides several methods to verify the integrity of transmitted messages and stored data:

- Message authentication code (MAC)
- Hash functions, including modification detection code (MDC) processing and one-way hash generation

The new callable services are:

- "MAC Generate2 (CSNBMGN2, CSNBMGN3, CSNEMGN2, and CSNEMGN3)"
- "MAC Verify2 (CSNBMVR2, CSNBMVR3, CSNEMVR2, and CSNEMVR3)" on page 11

How MACs are Used

When a message is sent, an application program can generate an authentication code for it using the MAC generation, MAC generation2, or HMAC generate callable service. ICSF supports:

- The ANSI X9.9-1 basic procedure and both the ANSI X9.19 basic procedure and optional double key MAC procedure for DES.
- Block cipher-based MAC algorithm, called CMAC (NIST SP 800-38B) for AES.
- FIPS-198 Keyed-Hash Message Authentication Code method for HMAC.

The message text may be in clear or encrypted form. The originator of the message sends the MAC with the message text.

MAC Generate2 (CSNBMGN2, CSNBMGN3, CSNEMGN2, and CSNEMGN3)

Use the MAC Generate2 callable service to generate a keyed hash message authentication code (HMAC) or a ciphered message authentication code (CMAC) for the message string provided as input. A MAC key with key usage that can be used for generate is required to calculate the MAC.

The MAC generate key must be in a variable-length HMAC key token for HMAC and an AES MAC token for CMAC.

The callable service names for AMODE(64) are CSNEMGN2 and CSNEMGN3.

Choosing between CSNBMGN2 and CSNBMGN3

CSNBMGN2 and CSNBMGN3 provide identical functions. When choosing which service to use, consider the following:

- CSNBMGN2 requires the application-supplied text to reside in the caller's primary address space.
- CSNBMGN3 allows the application-supplied text to reside either in the caller's primary address space or in a data space. This allows you to process more data with one call. For CSNBMGN3, *text_id_in* is an access list entry token (ALET) parameter of the data space containing the application-supplied text.

Format

```
CALL CSNBMGN2(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
```

```
                rule_array_count,
                rule_array,
                key_identifier_length,
                key_identifier,
                text_length,
                text,
                chaining_vector_length,
                chaining_vector,
                mac_length,
                mac )
CALL CSNBMGN3(
                return_code,
                reason_code,
                exit_data_length,
                exit_data,
                rule_array_count,
                rule_array,
                key_identifier_length,
                key_identifier,
                text_length,
                text,
                chaining_vector_length,
                chaining_vector,
                mac_length,
                mac,
                text_id_in )
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The 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, 2, or 3.

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 2. Keywords for MAC Generate2 Control Information

Keyword	Meaning
Token algorithm (One required)	
AES	Specifies the use of the AES CMAC algorithm to generate a MAC.
HMAC	Specifies the use of the HMAC algorithm to generate a MAC.
Hash method (One required for HMAC only)	
SHA-1	Specifies the use of the SHA-1 hash method.
SHA-224	Specifies the use of the SHA-224 hash method.
SHA-256	Specifies the use of the SHA-256 hash method.
SHA-384	Specifies the use of the SHA-384 hash method.
SHA-512	Specifies the use of the SHA-512 hash method.
Segmenting Control (One optional)	
FIRST	First call, this is the first segment of data from the application program.
LAST	Last call; this is the last data segment.
MIDDLE	Middle call; this is an intermediate data segment.
ONLY	Only call; segmenting is not employed by the application program. This is the default value.

key_identifier_length

Direction	Type
Input	Integer

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

key_identifier

Direction	Type
Input/Output	String

The identifier of the key to generate the MAC. The *key identifier* is an operational token or the key label of an operational token in key storage.

For the HMAC algorithm, the key algorithm must be HMAC and the key usage fields must indicate GENONLY or GENERATE and the hash method selected. For the AES algorithm, the key algorithm must be AES, the key type must be MAC, and the key usage fields must indicate GENONLY or GENERATE and must indicate CMAC.

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

text_length

Direction	Type
Input	Integer

The length of the text you supplied in the *text* parameter. The maximum length of *text* is 214783647 bytes. For FIRST and MIDDLE calls, the *text_length* must be:

- A multiple of 64 for the SHA-1, SHA-224, and SHA-256 hash methods.
- A multiple of 128 for the SHA-384 and SHA-512 hash methods.
- A multiple of 16 for the AES CMAC method.

text

Direction	Type
Input	String

The application-supplied text for which the MAC is generated.

chaining_vector_length

Direction	Type
Input/Output	Integer

chaining_vector_length specifies the length in bytes of the *chaining_vector* parameter. The value must be 128.

chaining_vector

Direction	Type
Input/Output	String

An 128-byte string that ICSF uses as a system work area. Your application program must not change the data in this string. The chaining vector permits data to be chained from one invocation call to another.

On the first call, initialize this parameter as binary zeros.

mac_length

Direction	Type
Input/Output	Integer

The length of the *mac* parameter in bytes. This parameter is updated to the actual length of the *mac* parameter on output. For HMAC, the minimum value is 4 and the maximum value is 64. For AES, the value must be 16.

mac

Direction	Type
Output	String

The field in which the callable service returns the MAC value if the segmenting rule is ONLY or LAST.

text_id_in

Direction	Type
Input	Integer

For CSNBMGN3 only, the ALET of the text for which the MAC is generated.

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 3. MAC Generate2 Access Control Points

Hash method	Access control point
CMAC	MAC Generate2 - AES CMAC
SHA-1	HMAC Generate - SHA-1
SHA-224	HMAC Generate - SHA-224
SHA-256	HMAC Generate - SHA-256
SHA-384	HMAC Generate - SHA-384
SHA-512	HMAC Generate - SHA-512

Required Hardware

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

Table 4. MAC 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.

Table 4. MAC Generate2 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	Requires the March 2014 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	Requires the March 2014 or later licensed internal code (LIC).

MAC Verify2 (CSNBMVR2, CSNBMVR3, CSNEMVR2, and CSNEMVR3)

Use the MAC Verify2 callable service to verify a keyed hash message authentication code (HMAC) or a ciphered message authentication code (CMAC) for the message text provided as input. A MAC key with key usage that can be used for verify is required to verify the MAC.

The MAC verify key must be in a variable-length HMAC key token for HMAC and an AES MAC token for CMAC.

The callable service names for AMODE(64) are CSNEMVR2 and CSNEMVR3.

Choosing between CSNBMVR2 and CSNBMVR3

CSNBMVR2 and CSNBMVR3 provide identical functions. When choosing which service to use, consider the following:

- CSNBMVR2 requires the application-supplied text to reside in the caller's primary address space.
- CSNBMVR3 allows the application-supplied text to reside either in the caller's primary address space or in a data space. This allows you to process more data with one call. For CSNBMVR3, *text_id_in* is an access list entry token (ALET) parameter of the data space containing the application-supplied text.

Format

```
CALL CSNBMVR2(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             text_length,
             text,
             chaining_vector_length,
             chaining_vector,
             mac_length,
             mac )
CALL CSNBMVR3(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
```

```
rule_array_count,
rule_array,
key_identifier_length,
key_identifier,
text_length,
text,
chaining_vector_length,
chaining_vector,
mac_length,
mac,
text_id_in )
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The 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, 2, or 3.

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 5. Keywords for MAC Verify2 Control Information

Keyword	Meaning
Token algorithm (One required)	
AES	Specifies the use of the AES CMAC algorithm to generate a MAC.
HMAC	Specifies the use of the HMAC algorithm to generate a MAC.
Hash method (One required for HMAC only)	
SHA-1	Specifies the use of the SHA-1 hash method.
SHA-224	Specifies the use of the SHA-224 hash method.
SHA-256	Specifies the use of the SHA-256 hash method.
SHA-384	Specifies the use of the SHA-384 hash method.
SHA-512	Specifies the use of the SHA-512 hash method.
Segmenting Control (One optional)	
FIRST	First call, this is the first segment of data from the application program.
LAST	Last call; this is the last data segment.
MIDDLE	Middle call; this is an intermediate data segment.
ONLY	Only call; segmenting is not employed by the application program. This is the default value.

key_identifier_length

Direction	Type
Input	Integer

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

key_identifier

Direction	Type
Input/Output	String

The identifier of the key to verify the MAC. The *key identifier* is an operational token or the key label of an operational token in key storage.

For the HMAC algorithm, the key algorithm must be HMAC and the key usage fields must indicate GENERATE or VERIFY and the hash method selected. For the AES algorithm, the key algorithm must be AES, the key type must be MAC, and the key usage fields must indicate GENERATE or VERIFY and must indicate CMAC.

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

text_length

Direction	Type
Input	Integer

The length of the text you supplied in the *text* parameter. The maximum length of *text* is 214783647 bytes. For FIRST and MIDDLE calls, the *text_length* must be:

- A multiple of 64 for the SHA-1, SHA-224, and SHA-256 hash methods.
- A multiple of 128 for the SHA-384 and SHA-512 hash methods.
- A multiple of 16 for the AES CMAC method.

text

Direction	Type
Input	String

The application-supplied text for which the MAC is generated.

chaining_vector_length

Direction	Type
Input/Output	Integer

chaining_vector_length specifies the length in bytes of the *chaining_vector* parameter. The value must be 128.

chaining_vector

Direction	Type
Input/Output	String

An 128-byte string that ICSF uses as a system work area. Your application program must not change the data in this string. The chaining vector permits data to be chained from one invocation call to another.

On the first call, initialize this parameter as binary zeros.

mac_length

Direction	Type
Input	Integer

The length of the *mac* parameter in bytes. For HMAC, the maximum value is 64. For AES, the value must be 16.

mac

Direction	Type
Input	String

The field that contains the MAC value you want to verify.

text_id_in

Direction	Type
Input	Integer

For CSNBMVR3 only, the ALET of the text for which the MAC is to be verified.

Usage Notes

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

Access Control Points

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

Table 6. MAC Verify2 Access Control Points

Hash method	Access control point
CMAC	MAC Verify2 - AES CMAC
SHA-1	HMAC Verify - SHA-1
SHA-224	HMAC Verify - SHA-224
SHA-256	HMAC Verify - SHA-256
SHA-384	HMAC Verify - SHA-384
SHA-512	HMAC Verify - SHA-512

Required Hardware

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

Table 7. MAC Verify2 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 March 2014 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	Crypto Express3 Coprocessor Crypto Express4 Coprocessor	Requires the March 2014 or later licensed internal code (LIC).

Financial Services for DK PIN Methods

This section provides information on financial services that are based on the PIN methods of and meet the requirements specified by the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)). DK is an association of the German banking industry. The intellectual property rights regarding the methods and specification belongs to the German Banking Industry Committee.

Note: All crypto coprocessors must be loaded with the same level of code. There have been several licensed internal code (LIC) released in support of the DK PIN methods. Ensure that all of the coprocessors have the same LIC level to support the function you want to use.

The callable services that support the German Banking Industry Committee (Deutsche Kreditwirtschaft (DK)) PIN methods are:

- "DK Deterministic PIN Generate (CSNBDDPG and CSNEDDPG)" on page 5
- "DK PAN Translate (CSNBDPT and CSNEDPT)" on page 5
- "DK PRW Card Number Update (CSNBDPNU and CSNEDPNU)" on page 5
- "DK PRW CMAC Generate (CSNBDPCG and CSNEDPCG)" on page 5
- "DK Regenerate PRW (CSNBDRP and CSNEDRP)" on page 5

Weak PIN table

The DK PIN methods support the use of 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 Deterministic PIN Generate (CSNBDDPG and CSNEDDPG)

Use the DK Deterministic PIN Generate callable service to generate a PIN and PIN reference value (PRW) using an AES PIN calculation key. The PIN reference value is used to verify the PIN in other services.

Note: If the generated PIN appears in the weak PIN table, the generation process is retried by appending to the account information until an acceptable PIN is generated. For additional information, see "Weak PIN table."

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 PRW 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.

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

Format

```
CALL CSFBDDPG(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             account_info_ER_length,
             account_info_ER,
             PAN_data_length,
             PAN_data,
             card_p_data_length,
             card_p_data,
             card_t_data_length,
             card_t_data,
             PIN_length,
             PIN_generation_key_identifier_length,
             PIN_generation_key_identifier,
             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.

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The 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	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. There are no keywords for this service.

Table 8. Rule array keywords for the DK Deterministic PIN 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 of the encrypted PIN block.

account_info_ER_length

Direction	Type
Input	Integer

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

account_info_ER

Direction	Type
Input	String

The 16-byte account information used to generate the PIN.

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	String

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.

PIN_generation_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *PIN_generation_key_identifier* parameter. If the *PIN_generation_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_generation_key_identifier

Direction	Type
Input/Output	String

The identifier of the PIN 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, the key type must be PINCALC, the key usage fields must indicate GENONLY, CBC, and DKPINOP.

If the token supplied was encrypted under the old master key, the token is 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 PRW 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, 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 is 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 is 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 is 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, and the key usage fields must indicate CMAC, GENONLY, and DKPINOP.

If the token supplied was encrypted under the old master key, the token is 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. The value 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 in 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 Deterministic 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 9. DK Deterministic 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.

Table 9. DK Deterministic PIN Generate 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 PAN Translate (CSNBDPT and CSNEDPT)

Use the DK PAN Translate callable service to create an encrypted PIN block with the same PIN and a different PAN. The account data may change, but changing the PIN is to be avoided. This service specifically creates a new encrypted PIN block and MAC on that encrypted PIN block, which will be used to accept the PAN change at an authorization node.

You can use this service to perform the following tasks:

- Generate an encrypted PIN block in PBF-1 format with a changed PAN to be used at the authorization node to create a PIN reference value.
- Generate a CMAC over the encrypted PIN block for validation.

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

Format

```
CALL CSNBDPT(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             card_p_data_length,
             card_p_data,
             card_t_data_length,
             card_t_data,
             new_PAN_data_length,
             new_PAN_data,
             new_card_p_data_length,
             new_card_p_data,
             PIN_reference_value_length,
             PIN_reference_value,
             PRW_random_number_length,
             PRW_random_number,
             current_encrypted_PIN_block_length,
             current_encrypted_PIN_block,
             current_PIN_block_MAC_length,
             current_PIN_block_MAC,
             PRW_key_identifier_length,
             PRW_key_identifier,
             IPIN_encryption_key_identifier_length,
             IPIN_encryption_key_identifier,
             IEPB_MAC_key_identifier_length,
             IEPB_MAC_key_identifier,
             OPIN_encryption_key_identifier_length,
```

```
OPIN_encryption_key_identifier,
OEPB_MAC_key_identifier_length,
OEPB_MAC_key_identifier,
new_encrypted_PIN_block_length,
new_encrypted_PIN_block,
new_PIN_block_MAC_length,
new_PIN_block_MAC)
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The 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	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. There are no keywords for this service.

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.

new_PAN_data_length

Direction	Type
Input	Integer

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

new_PAN_data

Direction	Type
Input	String

The new personal account number (in character form) which the PIN will be associated. The full account number, including check digit, should be included.

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), determined by the card issuer, which is used to differentiate between multiple cards for one account.

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.

current_encrypted_PIN_block_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *current_encrypted_PIN_block* parameter. The value must be 32.

current_encrypted_PIN_block

Direction	Type
Input	String

The 32-byte encrypted PIN block in PBF-1 format of the current PIN.

current_PIN_block_MAC_length

Direction	Type
Input	Integer

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

current_PIN_block_MAC

Direction	Type
Input	String

The 8-byte MAC of the current encrypted PIN block and the *card_p_data*.

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 PRW verifying 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, the key type must be PINPRW, the key usage fields must indicate VERIFY, CMAC, and DKPINOP.

If the token supplied was encrypted under the old master key, the token is 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 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 AES, the key type must be PINPROT, and the key usage fields must indicate DECRYPT, CBC, and DKPINOP.

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

IEPB_MAC_key_identifier_length

Direction	Type
Input	Integer

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

IEPB_MAC_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to verify MAC of the inbound 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 AES, the key type must be MAC, and the key usage fields must indicate CMAC. VERIFY, and DKPINOP.

If the token supplied was encrypted under the old master key, the token is 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 *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. 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 DKPINAD1.

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

OEPB_MAC_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *new_OEPB_MAC_key_identifier* parameter. If the *new_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 new 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 AES, the key type must be MAC, and the key usage fields must indicate CMAC, GENONLY, and DKPINAD1.

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

new_encrypted_PIN_block_length

Direction	Type
Input/Output	Integer

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

new_encrypted_PIN_block

Direction	Type
Output	String

The 32-byte encrypted new PIN block.

new_PIN_block_MAC_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *new_PIN_block_MAC* parameter. The value must be at least 8.

new_PIN_block_MAC

Direction	Type
Output	String

The 8-byte MAC of the new encrypted PIN block.

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 Translate** 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 10. 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.
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 PRW Card Number Update (CSNBDPNU and CSNEDPNU)

Use the DK PRW Card Number Update callable service to generate a PIN reference value (PRW) when a replacement card is being issued. The original PIN and primary account number are used with new time-sensitive card data to generate the new PRW.

You can use this service to perform the following tasks:

- Generate a PRW that 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.

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

Format

```
CALL CSNBDPNU(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             card_p_data_length,
             card_p_data,
             card_t_data_length,
             card_t_data,
             encrypted_PIN_block_length,
             encrypted_PIN_block,
             PIN_block_MAC_length,
             PIN_block_MAC,
             PRW_key_identifier_length,
             PRW_key_identifier,
```

```
IPIN_encryption_key_identifier_length,
IPIN_encryption_key_identifier,
IEPB_MAC_key_identifier_length,
IEPB_MAC_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,
new_encrypted_PIN_block_length,
new_encrypted_PIN_block,
new_PIN_block_MAC_length,
new_PIN_block_MAC)
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The 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	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 11. Keywords for the DK PRW Card Number Update service

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

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.

encrypted_PIN_block_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *encrypted_PIN_block* parameter. The value must be 32.

encrypted_PIN_block

Direction	Type
Input	String

The 32-byte input encrypted PIN block in PBF-1 format.

PIN_block_MAC_length

Direction	Type
Input	Integer

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

PIN_block_MAC

Direction	Type
Input	String

The 8-byte CMAC of the encrypted PIN block.

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 PRW 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, 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 is 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 that encrypts the input 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 PINPROT, and the key usage fields must indicate DECRYPT, CBC, and DKPINOP.

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

IEPB_MAC_key_identifier_length

Direction	Type
Input	Integer

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

IEPB_MAC_key_identifier

Direction	Type
Input/Output	String

The identifier of the CMAC verification 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, the key type must be MAC, and the key usage fields must indicate CMAC, VERIFY, and DKPINOP.

If the token supplied was encrypted under the old master key, the token is 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 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 new PIN block. The key identifier is an operational token or the key label of an operational token in key storage. If no encrypted PIN block is to be returned, this value 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 is 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 CMAC of the new PRW. 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 MAC is to be returned, 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 GENONLY, CMAC, and DKPINOP.

If the token supplied was encrypted under the old master key, the token is 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. This value must be 16. On output, it will be set to 16.

PIN_reference_value

Direction	Type
Output	String

The calculated 16-byte 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 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.

new_encrypted_PIN_block_length

Direction	Type
Input/Output	Integer

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

new_encrypted_PIN_block

Direction	Type
Output	String

The new 32-byte encrypted PIN block. If the rule array indicates that no new encrypted PIN block should be returned, this parameter is ignored.

new_PIN_block_MAC_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *new_PIN_block_MAC* parameter. If the *rule_array* indicates that no new_PIN_block_MAC should be returned, this value must be zero. Otherwise, it must be at least 8.

new_PIN_block_MAC

Direction	Type
Output	String

The new 8-byte encrypted MAC of the new PIN block. If the rule array indicates that no new encrypted PIN block should be returned, 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 PRW Card Number Update** 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 12. 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 12. 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 PRW CMAC Generate (CSNBDPCG and CSNEDPCG)

Use the DK PRW CMAC Generate callable service to generate a message authentication code (MAC) over specific values involved in an account number change transaction. The inputs include the current and new PAN and card data and the PIN reference value.

The output of this service is used as input to the DK PAN Modify in Transaction callable service, which will create the new PIN reference value (PRW) to be used to verify the PIN.

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

Format

```
CALL CSNBDPCG(
             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_data_length,
             current_card_data,
             new_card_data_length,
             new_card_data,
             PIN_reference_value_length,
             PIN_reference_value,
             CMAC_FUS_key_identifier_length,
             CMAC_FUS_key_identifier,
             CMAC_FUS_length,
             CMAC_FUS)
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The 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	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. 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	String

The current PAN data. The full account number, including check digit, should be included.

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	String

The new PAN data. The full account number, including check digit, should be included.

current_card_data_length

Direction	Type
Input	Integer

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

current_card_data

Direction	Type
Input	String

The current card data, determined by the card issuer.

new_card_data_length

Direction	Type
Input	Integer

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

new_card_data

Direction	Type
Input	String

The new card data, determined by the card issuer.

PIN_reference_value_length

Direction	Type
Input	Integer

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

PIN_reference_value

Direction	Type
Input	String

The 16-byte PIN reference value of the current PIN.

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	String

The identifier of the key to generate the MAC. 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 GENONLY, 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.

CMAC_FUS_length

Direction	Type
Input/Output	Integer

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

CMAC_FUS

Direction	Type
Output	String

The MAC of the current and new PANs and card data strings.

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 PRW CMAC 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 13. DK PRW CMAC 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 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 Regenerate PRW (CSNBDRP and CSNEDRP)

Use the DK Regenerate PRW callable service to generate a new PIN reference value for a changed account number.

You can use this service to perform the following tasks:

- Generate a PIN reference value over the existing PIN and new PAN, which can be used to verify transactions.
- Generate an encrypted PIN block in PBF-1 format to be stored for later use in personalization of smart cards.

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

Format

```
CALL CSNBDRP(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             card_p_data_length,
             card_p_data,
             card_t_data_length,
             card_t_data,
             encrypted_PIN_block_length,
             encrypted_PIN_block,
             PIN_block_MAC_length,
             PIN_block_MAC,
             PRW_key_identifier_length,
             PRW_key_identifier,
             IPIN_encryption_key_identifier_length,
             IPIN_encryption_key_identifier,
             IEPB_MAC_key_identifier_length,
             IEPB_MAC_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,
new_encrypted_PIN_block_length,
new_encrypted_PIN_block,
new_PIN_block_MAC_length,
new_PIN_block_MAC)
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Input/Output	Integer

The length of the data that is passed to the installation exit. The 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	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. There are no keywords for this service.

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.

encrypted_PIN_block_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *encrypted_PIN_block* parameter. The value must be 32.

encrypted_PIN_block

Direction	Type
Input	String

The 32-byte encrypted PIN block in PBF-1 format of the input PIN.

PIN_block_MAC_length

Direction	Type
Input	Integer

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

PIN_block_MAC

Direction	Type
Input	String

The 8-byte MAC of the encrypted PIN block.

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 PRW 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, 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 is 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 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 AES, the key type must be PINPROT, and the key usage fields must indicate DECRYPT, CBC, and DKPINAD1.

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

IEPB_MAC_key_identifier_length

Direction	Type
Input	Integer

Specifies the length in bytes of the *IEPB_MAC_key_identifier* parameter. If the *IEPB_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.

IEPB_MAC_key_identifier

Direction	Type
Input/Output	String

The identifier of the key to verify MAC of the inbound 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 AES, the key type must be MAC, and the key usage fields must indicate CMAC, VERIFY, and DKPINAD1.

If the token supplied was encrypted under the old master key, the token is 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 *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. 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 is 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 *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 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 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 is 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. This value must be 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 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.

new_encrypted_PIN_block_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *new_encrypted_PIN_block* parameter. The value should be at least 32.

new_encrypted_PIN_block

Direction	Type
Output	String

The 32-byte encrypted PIN block.

new_PIN_block_MAC_length

Direction	Type
Input/Output	Integer

Specifies the length in bytes of the *new_PIN_block_MAC* parameter. The value must be at least 8.

new_PIN_block_MAC

Direction	Type
Output	String

The 8-byte MAC of the encrypted PIN block.

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 Regenerate PRW** 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 14. DK Regenerate PRW 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).

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 15. Access control points – Callable Services

Name	Callable Service	Usage
DK Deterministic PIN Generate	CSNBDDPG / CSNEDDPG	DD
DK PAN Translate	CSNBDPT / CSNEDPT	DD
DK PRW Card Number Update	CSNBPNU / CSNEPNU	DD
DK PRW CMAC Generate	CSNBDPCG / CSNBPCG	DD
DK Regenerate PRW	CSNBDRP / CSNEDRP	DD
MAC Generate2 – AES CMAC	CSNBMGN2 / CSNEMGN2 / CSNBMGN3 / CSNEMGN3	ED
MAC Verify2 – AES CMAC	CSNBMVR2 / CSNEMVR2 / CSNBMVR3 / CSNEMVR3	ED

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 Part 2 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 16. Resource names for ICSF Callable Services

Resource Name	Callable Service Name(s)	Callable Service Description
CSFDPCG	CSNBDPCG CSNEDPCG	DK PRW CMAC Generate
CSFDDPG	CSNBDDPG CSNEDDPG	DK Deterministic PIN Generate
CSFDPNU	CSNBDPNU CSNEDPNU	DK PRW Card Number Update
CSFDPT	CSNBDPT CSNEDPT	DK PAN Translate
CSFDRP	CSNBDRP CSNEDRP	DK Regenerate PRW
CSFMGN2	CSNBMGN2 CSNEMGN2	MAC Generate2
CSFMGN3	CSNBMGN3 CSNEMGN3	MAC Generate2 with ALET
CSFMVR2	CSNBMVR2 CSNEMVR2	MAC Verify2
CSFMVR3	CSNBMVR3 CSNEMVR3	MAC Verify2 with ALET

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.

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 1 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 1. 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.

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.
- 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 17. 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.

Table 17. Key types (continued)

Key Type	Algorithm	Usage	Notes
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.
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

Table 17. Key types (continued)

Key Type	Algorithm	Usage	Notes
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 18 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 18. 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
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 19. Usage values for key types

Key type	Key algorithm	Key Usage Values
CIPHER	AES	The following values are optional: C-XLATE, V1PYLD and One or both may be specified: DECRYPT, ENCRYPT. **Note:** The key generated when KEYUSAGE is not specified will have only the DECRYPT and ENCRYPT key-usage. This is the default.
DKYGENKY	DES	One of the following must be specified: DKYL0, DKYL1, DKYL2, DKYL3, DKYL4, DKYL5, DKYL6, DKYL7 and One of the following must be specified: DALL, DDATA, DEXP, DIMP, DMAC, DMKEY, DMPIN, DMV, DPVR
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 values must be specified: KUF-MBE, KUF-MBP
DKYGENKY	AES	The following values are required: D-CIPHER, DKYL0 and The following value is optional: DKYUSAGE and One of the following values may be specified when DKYUSAGE is specified: KUF-MBE, KUF-MBP (KUP-MBE is the default)

Table 19. Usage values for key types (continued)

Key type	Key algorithm	Key Usage Values
DKYGENKY	AES	One of the following must be specified: D-ALL, 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.
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 20. 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.
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.

Table 20. Meaning of usage values (continued)

Key Usage Value	Key types	Meaning
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.
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).

Table 20. Meaning of usage values (continued)

Key Usage Value	Key types	Meaning
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 21. The other values do not have a complementary value and are copied.

Table 21. Complementary key-usage values for AES CIPHER

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

Table 21. Complementary key-usage values for AES CIPHER (continued)

Key usage values	Complementary key usage values
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 22. 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 22. 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 23 on page 66 and Table 24 on page 66. 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 23. Values by type for DKYGENKYUSAGE

Type of key to be derived	DKYGENKYUSGE values
CIPHER	The following values are optional: C-XLATE, DECRYPT, ENCRYPT **Note:** The key generated when DKYGENKYUSAGE is not specified will have DECRYPT and ENCRYPT key-usage. This is the default.
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.
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 24. 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.
C-XLATE	CIPHER	Restricts the key to be used with the cipher text translate2 service only.

Table 24. Meaning of usage values (continued)

Value	Key types	Description
DECRYPT	CIPHER, PINPROT	The derived key may be used to decrypt PIN blocks.
DKPINAD1	MAC, PINPROT	The derived key may be used to create or verify a pin block to allow changing the account number associate with a PIN for the DK PIN methods.
DKPINAD2	MAC	The derived key may be used to create or verify an account change string to allow changing the account number associated with a PIN for the DK PIN methods.
DKPINOP	MAC, PINCALC, PINPROT, PINPRW	The derived key may be used as a general purpose key for the DK PIN methods.
DKPINOPP	PINPROT	The derived key may be used to encrypt a PIN block for the specific purpose of creating a PIN mailer for the DK PIN methods.
ENCRYPT	CIPHER, PINPROT	The derived key may be used to encrypt PIN blocks.
GENERATE	MAC	The derived key may be used to generate and verify MACs.
GENONLY	MAC, PINCALC	The derived key may be used to generate MACs or PINs.
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 25. Values that do not appear in the table are copied for the complementary key.

Table 25. Complementary values for usage values

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

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

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

Callable services affected by key store policy

This table provides application programmers guidance on parameters covered by the key store policy controls.

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

Table 26. Callable services and parameters affected by key store policy

ICSF callable service	31-bit name	Parameter checked
DK Deterministic PIN Generate	CSNBDDPG	PIN_generation_key_identifier PRW_key_identifier PIN_print_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PAN Translate	CSNBDPT	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PRW Card Number Update	CSNBDPNU	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
DK PRW CMAC Generate	CSNBDPCG	CMAC_FUS_key_identifier
DK Regenerate PRW	CSNBDRP	PRW_key_identifier IPIN_encryption_key_identifier IEPB_MAC_key_identifier OPIN_encryption_key_identifier OEPB_MAC_key_identifier
MAC Generate2	CSNBMGN2	key_identifier
MAC Generate2 with ALET	CSNBMGN3	key_identifier
MAC Verify2	CSNBMVR2	key_identifier
MAC Verify2 with ALET	CSNBMVR3	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 Part 2 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

Table 27. Exit identifiers and exit invocations

Exit identifiers	Exit invocations
CSFDDPG	Gets control during the DK Deterministic PIN Generate callable service.
CSFDPCG	Gets control during the DK PRW CMAC Generate callable service.
CSFDPNU	Gets control during the DK PRW Card Number Update callable service.
CSFDPT	Gets control during the DK PAN Translate callable service.
CSFDRP	Gets control during the DK Regenerate PRW callable service.
CSFMGN2	Gets control during the MAC Generate2 callable service.
CSFMGN3	Gets control during the MAC Generate3 callable service.
CSFMVR2	Gets control during the MAC Verify2 callable service.
CSFMVR3	Gets control during the MAC Verify3 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

These topics describe common activities and considerations that should be considered when you migrate from an earlier release of ICSF to FMID HCR77A1 or HCR77A0.

Callable Services

The following table summarizes the new and changed callable services for ICSF FMID HCR77A1 and HCR77A0. For complete reference information on these callable services, refer to *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Table 28. Summary of new and changed ICSF callable services

Callable service	Release	Description
DK Deterministic PIN Generate	HCR77A0	**New:** Generate a PIN using a secret key.

Table 28. Summary of new and changed ICSF callable services (continued)

Callable service	Release	Description
DK PAN Translate	HCR77A0	**New:** Modify the PAN of an account while keeping the same PIN.
DK PRW Card Number Update	HCR77A0	**New:** Generate a PIN reference value (PRW) when a replacement card is being issued.
DK PRW CMAC Generate	HCR77A0	**New:** Generate a message authentication code (MAC) over specific values involved in an account number change transaction.
DK Regenerate PRW	HCR77A0	**New:** Generate a new PIN reference value for a changed account number.
MAC Generate2	HCR77A0	**New:** Generate a MAC using AES or HMAC keys.
MAC Verify2	HCR77A0	**New:** Verify a MAC using AES or HMAC keys.

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:

- ICSF FMID HCR77A1 only:
  - HCR77A1: CSFAPG, CSFPFO, CSFSXD
- ICSF FMID HCR77A1 and HCR77A0:
  - HCR77A0: CSFCTT2, CSFCTT3, CSFUDK, CSFDPV, CSFDPC, CSFDPMT, CSFDRPG, CSFDKG2, CSFDDPG, CSFDPCG, CSFDPNU, CSFDPT, CSFDRP, CSFMGN2, CSFMGN3, CSFMVR2, CSFMVR3
  - 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.

Resource Manager Interface (RMF)

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

- Decipher (CSNBDEC)
- Digital Signature Generate (CSNDDSG)
- Digital Signature Verify (CSNDDSV)
- Encipher (CSNBENC)
- MAC Generate (CSNBMGN)

- MAC Generate2 (CSNBMGN2)
- MAC Verify (CSNBMVR)
- MAC Verify2 (CSNBMVR2)
- One-Way Hash (CSNBOWH)
- PIN Translate (CSNBPTR)
- PIN Verify (CSNBPVR)
- Symmetric Algorithm Decipher (CSNBSAD)
- Symmetric Algorithm Encipher (CSNBSAE)

Diagnosis Reference Information

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

RMF measurements table

Table 29 describes the contents of the performance measurements for RMF. The count fields are double-word length.

Table 29. RMF measurements record format

Offset (Dec)	Number of bytes	Field name	Description
0	4	DACC_ID	The DACC ID.
4	4	DACC_VER	The version.
8	4	DACC_LEN	The control block length.
12	2	DACC_ENT_CNT	Number of entries.
14	2	DACC_ENT_LEN	Length of each entry.
16	8	DACC_ENT_ID	Identifier of count array - character 'ENCSDES'. The Encipher service will collect data as follows: • Collection for single DES is done separately. The number of service calls, number of bytes of data enciphered, and the number of hardware instructions used to encipher the data will be collected.
24	8	DACC_ENT_SVC_CNT	Count of ENCSDES service calls.
32	8	DACC_ENT_BYT_CNT	Count of ENCSDES bytes processed.
40	8	DACC_ENT_INT_CNT	Count of ENCSDES instructions.
48	8	DACC_ENT_ID	Identifier of count array - character 'ENCTDES'. The Encipher service will collect data as follows: • Double and triple DES will be counted together. The number of service calls, number of bytes of data enciphered, and the number of hardware instructions used to encipher the data will be collected.
56	8	DACC_ENT_SVC_CNT	Count of ENCTDES service calls.
64	8	DACC_ENT_BYT_CNT	Count of ENCTDES bytes processed.
72	8	DACC_ENT_INT_CNT	Count of ENCTDES instructions.

Table 29. RMF measurements record format (continued)

Offset (Dec)	Number of bytes	Field name	Description
80	8	DACC_ENT_ID	Identifier of count array - character DECSDES. The Decipher service will collect data as follows: • Collection for single DES is done separately. The number of service calls, number of bytes of data deciphered, and the number of hardware instructions used to decipher the data will be collected.
88	8	DACC_ENT_SVC_CNT	Count of DECSDES service calls.
96	8	DACC_ENT_BYT_CNT	Count of DECSDES bytes processed.
104	8	DACC_ENT_INT_CNT	Count of DECSDES instructions.
112	8	DACC_ENT_ID	Identifier of count array - character DECTDES. The Decipher service will collect data as follows: • Double and triple DES will be counted together. The number of service calls, number of bytes of data deciphered, and the number of hardware instructions used to decipher the data will be collected.
120	8	DACC_ENT_SVC_CNT	Count of DECTDES service calls.
128	8	DACC_ENT_BYT_CNT	Count of DECTDES bytes processed.
136	8	DACC_ENT_INT_CNT	Count of DECTDES instructions.
144	8	DACC_ENT_ID	Identifier of count array - character MACGEN. The MAC Generate service will collect data as follows: • Single and various double key MAC will be gathered together. The number of service calls, number of bytes of data MAC'd, and the number of instructions will be collected.
152	8	DACC_ENT_SVC_CNT	Count of MACGEN service calls.
160	8	DACC_ENT_BYT_CNT	Count of MACGEN bytes processed.
168	8	DACC_ENT_INT_CNT	Count of MACGEN instructions.
176	8	DACC_ENT_ID	Identifier of count array - character MACVER. The MAC Verify service will collect data as follows: • Single and various double key MAC will be gathered together. The number of service calls, number of bytes of data MAC'd, and the number of instructions will be collected.
184	8	DACC_ENT_SVC_CNT	Count of MACVER service calls.
192	8	DACC_ENT_BYT_CNT	Count of MACVER bytes processed.
200	8	DACC_ENT_INT_CNT	Count of MACVER instructions.
208	8	DACC_ENT_ID	Identifier of count array - character OWH. The One Way Hash service will collect data as follows: • For SHA-1, the number of service calls, number of bytes of bytes of data hashed, and the number of instructions will be collected.
216	8	DACC_ENT_SVC_CNT	Count of OWH service calls.
224	8	DACC_ENT_BYT_CNT	Count of OWH bytes processed.
232	8	DACC_ENT_INT_CNT	Count of OWH instructions.

Table 29. RMF measurements record format (continued)

Offset (Dec)	Number of bytes	Field name	Description
240	8	DACC_ENT_ID	Identifier of count array - character PTR. The PIN Translate service will collect data as follows: • Collect the number of service calls only.
248	8	DACC_ENT_SVC_CNT	Count of PTR service calls.
256	16		Reserved.
272	8	DACC_ENT_ID	Identifier of count array - character PVR. The PIN Verify service will collect data as follows: • Collect the number of service calls only.
280	8	DACC_ENT_SVC_CNT	Count of PVR service calls.
288	16		Reserved.
304	8	DACC_ENT_ID	Identifier of count array - character OWH256. The One Way Hash service will collect data as follows: • For SHA-224 and SHA-256, the number of service calls, number of bytes of data hashed, and the number of instructions will be collected.
312	8	DACC_ENT_SVC_CNT	Count of OWH service calls for SHA-224 and SHA-256.
320	8	DACC_ENT_BYT_CNT	Count of OWH bytes processed for SHA-224 and SHA-256.
328	8	DACC_ENT_INT_CNT	Count of OWH instructions for SHA-224 and SHA-256.
336	8	DACC_ENT_ID	Identifier of count array - character OWH512. The One Way Hash service will collect data as follows: • For SHA-384 and SHA-512, the number of service calls, number of bytes of data hashed, and the number of instructions will be collected.
344	8	DACC_ENT_SVC_CNT	Count of OWH service calls for SHA-384 and SHA-512.
352	8	DACC_ENT_BYT_CNT	Count of OWH bytes processed for SHA-384 and SHA-512.
360	8	DACC_ENT_INT_CNT	Count of OWH instructions for SHA-384 and SHA-512.
368	8	DACC_ENT_ID	Identifier of count array - character 'ENCAES'. The Symmetric algorithm encipher service will collect data as follows: The number of service calls, number of bytes of data enciphered, and the number of instructions used to encipher the data will be collected.
376	8	DACC_ENT_SVC_CNT	Count of SAE service calls
384	8	DACC_ENT_BYT_CNT	Count of ENCAES bytes processed
392	8	DACC_ENT_INT_CNT	Count of ENCAES instruction
400	8	DACC_ENT_ID	Identifier of count array - character 'DECAES'. The Symmetric algorithm decipher service will collect data as follows: the number of service calls, number of bytes of data deciphered, and the number of instructions used to decipher the data will be collected.
408	8	DACC_ENT_SVC_CNT	Count of SAD service calls
416	8	DACC_ENT_BYT_CNT	Count of DECAES bytes processed
424	8	DACC_ENT_INT_CNT	Count of DECAES instruction

Table 29. RMF measurements record format (continued)

Offset (Dec)	Number of bytes	Field name	Description
432	8	DACC_ENT_ID	Identifier of count array - character 'DSGRSA'. The Digital Signature Generate service will collect the number of service calls processed to generate a digital signature using an RSA private key.
440	8	DACC_ENT_SVC_CNT	Count of DSG service calls using an RSA private key
448	16		Reserved
464	8	DACC_ENT_ID	Identifier of count array - character 'DSGECC'. The Digital Signature Generate service will collect the number of service calls processed to generate a digital signature using an ECC private key.
472	8	DACC_ENT_SVC_CNT	Count of DSG service calls using an ECC private key
480	16		Reserved
496	8	DACC_ENT_ID	Identifier of count array - character 'DSVRSA'. The Digital Signature Verify service will collect the number of service calls processed to verify a digital signature using an RSA private key.
504	8	DACC_ENT_SVC_CNT	Count of DSV service calls using an RSA private key
512	16		Reserved
528	8	DACC_ENT_ID	Identifier of count array - character 'DSVECC'. The Digital Signature Verify service collects the number of service calls processed to verify a digital signature using an ECC private key.
536	8	DACC_ENT_SVC_CNT	Count of DSV service calls using an ECC private key
544	16		Reserved
560	8	DACC_ENT_ID	Identifier of count array - character 'MACGEN2'. The MAC Generate2 service collects data as follows: • The number of service calls. • The number of bytes of data MACed. • The number of instructions used to MAC the data.
568	8	DACC_ENT_SVC_CNT	Count of MACGEN2 service calls.
576	8	DACC_ENT_BYT_CNT	Count of MACGEN2 bytes processed.
584	8	DACC_ENT_INT_CNT	Count of MACGEN2 instructions.
592	8	DACC_ENT_ID	Identifier of count array - character 'MACVER2'. The MAC Verify2 service collects data as follows: • The number of service calls. • The number of bytes of data MACed. • The number of instructions used to MAC the data.
600	8	DACC_ENT_SVC_CNT	Count of MACVER2 service calls.
608	8	DACC_ENT_BYT_CNT	Count of MACVER2 bytes processed.
616	8	DACC_ENT_INT_CNT	Count of MACVER2 instructions.

Chapter 5. Update of z/OS Cryptographic Services ICSF Overview, SC14-7505-00, information

This topic contains updates to the document *z/OS Cryptographic Services ICSF Overview*, SC14-7505-00, for the DK AES PIN Part 2 support provided by this APAR. Refer to this source document if background information is needed.

Standards

The Cryptographic Coprocessor Feature, PCI Cryptographic Coprocessor, and ICSF provide support for these International and USA standards (at least in part):

NIST SP 800-38B Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005

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