z/OS

Cryptographic Services Integrated Cryptographic Service Facility AES Counter Mode Support - APAR OA45548

Contents

Chapter 1. Overview

This document describes changes to the Integrated Cryptographic Service Facility (ICSF) product in support of the counter (CTR) mode for the AES algorithm. Support for AES CTR mode is added to:

- Symmetric Key Decipher (CSNBSYD or CSNBSYD1 and CSNESYD or CSNESYD1)
- Symmetric Key Encipher (CSNBSYE or CSNBSYE1 and CSNESYE or CSNESYE1)
- PKCS #11 Secret key decrypt (CSFPSKD and CSFPSKD6)
- PKCS #11 Secret key encrypt (CSFPSKE and CSFPSKE6)

These changes are available through the application of the PTF for APAR OA45548 and apply to FMID HCR77A1, HCR77A0, HCR7790, and HCR7780.

This document contains alterations to information previously presented in *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SC14-7508-01.

The technical changes made to the ICSF product by the application of the PTF for APAR OA45548 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, SC14-7508-01, information

This topic contains updates to the document, *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SC14-7508-01, for the AES counter (CTR) mode support provided by this APAR. Refer to this source document if background information is needed.

Symmetric Key Decipher (CSNBSYD or CSNBSYD1 and CSNESYD or CSNESYD1)

Use the symmetric key decipher callable service to decipher data using one of the supported modes. ICSF supports several processing rules to decipher data. You choose the type of processing rule that the Symmetric Key Decipher callable service should use for block chaining.

Processing Rule
Purpose

ANSI X9.23
For cipher block chaining. The ciphertext must be an exact multiple of the block size for the specified algorithm (8 bytes for DES). The plaintext will be between 1 and 8 bytes shorter than the ciphertext. This process rule always pads the plaintext during encryption so that ciphertext produced is an exact multiple of the block size, even if the plaintext was already a multiple of the blocksize.

CBC For cipher block chaining. The ciphertext must be an exact multiple of the block size for the specified algorithm (8 bytes for DES, 16 bytes for AES). The plaintext will have the same length as the ciphertext.

CBC-CS
For cipher block chaining. The ciphertext must be at least the block size for the specified algorithm (8 bytes for DES, 16 bytes for AES). The plaintext will have the same length as the ciphertext.

CFB Performs cipher feedback encryption with the segment size equal to the block size. The ciphertext can be of any length. The plaintext will have the same length as the ciphertext.

CFB-LCFB
Performs cipher feedback encryption with the segment size set by the caller. The ciphertext can be of any length. The plaintext will have the same length as the ciphertext.

CTR Performs counter mode decryption. The ciphertext can be any length. The plaintext will have the same length as the ciphertext.

CUSP For cipher block chaining. The ciphertext can be of any length. The plaintext will have the same length as the ciphertext.

ECB Performs electronic code book encryption. The ciphertext must be an exact multiple of the block size for the specified algorithm (8 bytes for DES, 16 bytes for AES). The plaintext will have the same length as the ciphertext.

GCM Perform Galois/Counter mode decryption, which provides both confidentiality and authentication for the plaintext and authentication for

the additional authenticated data (AAD). The ciphertext can be any length. The plaintext will have the same length as the ciphertext. Additionally, the authentication tag will be verified before any data is returned.

IPS For cipher block chaining. The ciphertext can be any length. The plaintext will have the same length as the ciphertext.

OFB Perform output feedback mode encryption. The ciphertext can be any length. The plaintext will have the same length as the ciphertext.

PKCS-PAD
For cipher block chaining. The ciphertext must be an exact multiple of the block size (8 bytes for DES and 16 bytes for AES). The plaintext will be between 1 and the blocksize (8 bytes for DES, 16 bytes for AES) bytes shorter than the ciphertext. This process rule always pads the ciphertext so that ciphertext produced is an exact multiple of the blocksize, even if the plaintext was already a multiple of the blocksize.

The Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are supported. AES encryption uses a 128-, 192-, or 256-bit key. DES encryption uses a 56-, 112-, or 168-bit key. See the processing rule descriptions for limitations. For each algorithm, certain processing rules are not allowed. See the rule_array parameter description for more information.

All modes except ECB use an initial chaining vector (ICV) in their processing.

All modes that utilize chaining produce a resulting chaining value called the output chaining vector (OCV). The application can pass the OCV as the ICV in the next decipher call. This results in record chaining.

The selection between single-DES decryption mode and triple-DES decryption mode is controlled by the length of the key supplied in the key_identifier parameter. If a single-length key is supplied, single-DES decryption is performed. If a double-length or triple-length key is supplied, triple-DES decryption is performed.

The key may be specified as a clear key value, an internal clear key token, or the label name of a clear key or an encrypted key in the CKDS.

Choosing between CSNBSYD and CSNBSYD1

CSNBSYD and CSNBSYD1 provide identical functions. When choosing which service to use, consider this:

- **CSNBSYD** requires the ciphertext and plaintext to reside in the caller's primary address space. Also, a program using CSNBSYD adheres to the IBM Common Cryptographic Architecture: Cryptographic Application Programming Interface.

  The callable service name for AMODE(64) invocation is CSNESYD.
- **CSNBSYD1** allows the ciphertext and plaintext to reside either in the caller's primary address space or in a data space. This can allow you to decipher more data with one call. However, a program using CSNBSYD1 does not adhere to the IBM Common Cryptographic Architecture: Cryptographic Application Programming Interface and may need to be modified prior to it running with other cryptographic products that follow this programming interface.

  For CSNBSYD1, *cipher_text_id* and *clear_text_id* are access list entry token (ALET) parameters of the data spaces containing the ciphertext and plaintext.

  The callable service name for AMODE(64) invocation is CSNESYD1.

Format

```
CALL CSNBSYD(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             key_parms_length,
             key_parms,
             block_size,
             initialization_vector_length,
             initialization_vector,
             chain_data_length,
             chain_data,
             cipher_text_length,
             cipher_text,
             clear_text_length,
             clear_text,
             optional_data_length,
             optional_data)
CALL CSNBSYD1(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             key_parms_length,
             key_parms,
             block_size,
             initialization_vector_length,
             initialization_vector,
             chain_data_length,
             chain_data,
             cipher_text_length,
             cipher_text,
             clear_text_length,
             clear_text,
             optional_data_length,
             optional_data
             cipher_text_id
             clear_text_id)
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Ignored	Integer

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

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	String

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

Table 1. Symmetric Key Decipher Rule Array Keywords

Keyword	Meaning
Algorithm (required)	
AES	Specifies that the Advanced Encryption Standard (AES) algorithm is to be used. The block size is 16 bytes. The key length may be 16, 24, or 32 bytes. The *chain_data* field must be at least 32 bytes in length. The OCV is the first 16 bytes in the *chain_data*. AES does not support the CUSP, IPS, or X9.23 processing rules.
DES	Specifies that the Data Encryption Standard (DES) algorithm is to be used. The algorithm, DES or TDES, will be determined from the length of the key supplied. The key length may be 8, 16, or 24. The block size is 8 bytes. The *chain_data* field must be at least 16 bytes in length. The OCV is the first eight bytes in the *chain_data*. DES does not support the CTR or GCM processing rules.
Processing Rule (optional) Rules CBC-CS, CUSP, IPS, PKCS-PAD, and X9.23 should be specified only when there is one request or on the last request of a sequence of chained requests.	

Table 1. Symmetric Key Decipher Rule Array Keywords (continued)

Keyword	Meaning
CBC	Performs cipher block chaining. The text length must be a multiple of the block size for the specified algorithm. CBC is the default value.
CBC-CS	CBC mode (cipher block chaining) with ciphertext stealing. The text length must be at least the block size for the specified algorithm.
CFB	CFB mode (cipher feedback) that is compatible with IBM's Encryption Facility product. Input text may be any length.
CFB-LCFB	CFB mode (cipher feedback). This rule allows the value of *s* (the segment size) to be something other than the block size (*s* is set to the block size with the CFB processing rule). *key_parms_length* and *key_parms* are used to set the value of *s*. Input text may be any length.
CTR	CTR mode (counter mode). Input text may be any length.
CUSP	CBC mode (cipher block chaining) that is compatible with IBM's CUSP and PCF products. Input text may be any length.
ECB	Performs electronic code book encryption. The text length must be a multiple of the block size for the specified algorithm.
GCM	GCM (Galois/Counter Mode). *key_parms_length* and *key_parms* are used to indicate the length of the tag (the value *t*) on input and contain the tag on output. Additional Authenticated Data (AAD) is contained in *optional_data_length* and *optional_data*. Input text may be any length. GCM does not support chaining, so CONTINUE and FINAL are not allowed for the ICV Selection rule.
IPS	CBC mode (cipher block chaining) that is compatible with IBM's IPS product. Input text may be any length.
OFB	OFB mode (output feedback). Input text may be any length.
PKCS-PAD	CBC mode (cipher block chaining) but the ciphertext must be an exact multiple of the block length (8 bytes for DES and 16 bytes for AES). The plaintext will be 1 to 8 bytes shorter for DES and 1 to 16 bytes shorter for AES than the ciphertext.
X9.23	CBC mode (cipher block chaining) for 1 to 8 bytes of padding dropped from the output clear text.
Key Rule (optional)	
KEY-CLR	This specifies that the key parameter contains a clear key value. KEY-CLR is the default value.
KEYIDENT	This specifies that the *key_identifier* field will be an internal clear token, or the label name of a clear key or encrypted key in the CKDS. Normal CKDS labelname syntax is required.
ICV Selection (optional)	
INITIAL	This specifies taking the initialization vector from the *initialization_vector* parameter. INITIAL is the default value. INITIAL is not valid with processing rule GCM.

Table 1. Symmetric Key Decipher Rule Array Keywords (continued)

Keyword	Meaning
CONTINUE	This specifies taking the initialization vector from the output chaining vector contained in the work area to which the *chain_data* parameter points. CONTINUE is not valid for processing rules ECB, GCM, or X9.23.
FINAL	This specifies taking the initialization vector from the output chaining vector contained in the work area to which the *chain_data* parameter points. Using FINAL indicates that this call contains the last portion of data. FINAL is valid for processing rules CBC-CS, CFB, CFB-LCBF, CTR, and OFB.
ONLY	This specifies taking the initialization vector from the *initialization_vector* parameter and that the entirety of the data to be processed is in this single call. ONLY is valid for processing rules CBC-CS, CFB, CFB-LCFB, CTR, GCM, and OFB.

key_identifier_length

Direction	Type
Input	Integer

The length of the *key_identifier* parameter. For clear keys, the length is in bytes and includes only the value of the key. The maximum size is 256 bytes.

For the KEYIDENT keyword, this parameter value must be 64.

key_identifier

Direction	Type
Input	String

For the KEY-CLR keyword, this specifies the cipher key. The parameter must be left justified.

For the KEYIDENT keyword, this specifies an internal clear token, or the label name of a clear key or an encrypted key in the CKDS. Normal CKDS label name syntax is required. The key algorithm may be DES or AES and the key type must be DATA. Encrypted key support is available on IBM System z10 and later servers.

key_parms_length

Direction	Type
Input	Integer

The length of the *key_parms* parameter.

- For the CFB-LCFB and CTR processing rules, this length must be 1.
- For the GCM processing rule, this is the length in bytes of the authentication tag to be verified. Valid lengths are 4, 8, 12, 13, 14, 15, 16. Using a length of 4 or 8 is strongly discouraged.
- For all other processing rules, this field is ignored.

You must specify the same length used when enciphering the text.

key_parms

Direction	Type
Input	String

This parameter contains key-related parameters specific to the encryption algorithm and processing mode.

- For the CFB-LCFB processing rule, this 1-byte field specifies the segment size in bytes. Valid values are 1 to the block size, inclusive. The block size is eight for DES and sixteen for AES.
- For the CTR processing rule, this 1-byte field specifies the number of low order bytes of the counter to be incremented. The remaining upper order bytes are the nonce. Valid values are 1 to the block size, inclusive. The blocksize is sixteen for AES.
- For the GCM processing rule, this contains the authentication tag for the provided ciphertext (*cipher_text* parameter) and additional authenticated data (*optional_data* parameter).
- For all other processing rules, this field is ignored.

For the modes where *key_parms* is used, you must specify the same *key_parms* used when enciphering the text using the Symmetric Key Encipher.

block_size

Direction	Type
Input	Integer

This parameter contains the processing size of the text block in bytes. This value will be algorithm specific. Be sure to specify the same block size as used to encipher the text.

initialization_vector_length

Direction	Type
Input	Integer

The length of the *initialization_vector* parameter. This parameter is ignored for the ECB processing rule. For the GCM processing rule, NIST recommends a length of 12, but tolerates any non-zero length. For all other processing rules, the length should be equal to the block length for the algorithm specified.

initialization_vector

Direction	Type
Input	String

This initialization chaining value. You must use the same ICV that was used to encipher the data. This parameter is ignored for the ECB processing rule.

chain_data_length

Direction	Type
Input/Output	Integer

The length of the *chain_data* parameter. On output, the actual length of the chaining vector will be stored in the parameter. This parameter is ignored if the ICV selection keyword is ONLY.

chain_data

Direction	Type
Input/Output	String

This field is used as a system work area for the chaining vector. Your application program must not change the data in this string. The chaining vector holds the output chaining vector from the caller.

The direction is output if the ICV selection keyword is INITIAL. This parameter is ignored if the ICV selection keyword is ONLY.

The mapping of the *chain_data* depends on the algorithm specified. For AES, the *chain_data* field must be at least 32 bytes in length. The OCV is in the first 16 bytes in the *chain_data*. For DES, *chain_data* field must be at least 16 bytes in length.

cipher_text_length

Direction	Type
Input	Integer

The length of the ciphertext. A zero value in the *cipher_text_length* parameter is not valid except with the GCM processing rule when performing a GMAC operation. The length must be a multiple of the algorithm block size for the CBC, ECB, and PKCS-PAD processing rules, but may be any length with the other processing rules.

cipher_text

Direction	Type
Input	String

The text to be deciphered.

clear_text_length

Direction	Type
Input/Output	Integer

On input, this parameter specifies the size of the storage pointed to by the *clear_text* parameter. On output, this parameter has the actual length of the text stored in the *clear_text* parameter. The *clear_text* parameter must be at least the same length as the *cipher_text* parameter, except for the PKCS-PAD and X9.23 processing rules, where the padding is automatically dropped on output.

clear_text

Direction	Type
Output	String

The deciphered text the service returns.

optional_data_length

Direction	Type
Input	Integer

The length of the *optional_data* parameter. For the GCM processing rule, this parameter contains the length of the Additional Authenticated Data (AAD). For all other processing rules, this field is ignored.

optional_data

Direction	Type
Input	String

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

You must specify the same *optional_data* used when enciphering the text using Symmetric Key Encipher.

cipher_text_id

Direction	Type
Input	Integer

For CSNBSYD1 only, the ALET of the ciphertext to be deciphered.

clear_text_id

Direction	Type
Input	Integer

For CSNBSYD1 only, the ALET of the clear text supplied by the application.

Usage notes

- SAF may be invoked to verify the caller is authorized to use the specified key label stored in the CKDS.
- To use a DES or AES encrypted DATA key in the CKDS, the ICSF segment of the CSFKEYS class general resource profile associated with the specified key label must contain SYMCPACFWRAP(YES).
- No pre- or post-processing exits are enabled for this service.
- The master keys need to be loaded only when using this service with encrypted key labels.
- The AES algorithm will use hardware if it is available. Otherwise, clear key operations will be performed in software.
- AES has the same availability restrictions as triple-DES.
- This service will fail if execution would cause destructive overlay of the *cipher_text* field.

Access control points

When the label of an encrypted key is specified for the *key_identifier* parameter, the appropriate access control point listed below must be enabled.

Symmetric Key Decipher

Table 2. Required access control points for Symmetric Key Decipher

Key algorithm	Access control point
AES	Symmetric Key Encipher/Decipher - Encrypted AES keys
DES	Symmetric Key Encipher/Decipher - Encrypted DES keys

Required hardware

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

Table 3. Symmetric Key Decipher required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890	CP Assist for Cryptographic Functions	GCM processing rule is not supported. CFB-LCFB processing rule is supported only when *key_parms* specifies a segment size equal to the blocksize.
IBM System z9 EC IBM System z9 BC	CP Assist for Cryptographic Functions	GCM processing rule is not supported. CFB-LCFB processing rule is supported only when *key_parms* specifies a segment size equal to the blocksize.
IBM System z10 EC IBM System z10 BC	CP Assist for Cryptographic Functions Crypto Express3 Coprocessor	GCM processing rule is not supported. CFB-LCFB processing rule is supported only when *key_parms* specifies a segment size equal to the blocksize. Encrypted keys require CEX3C with the Nov. 2009 or later licensed internal code (LIC).
IBM zEnterprise 196 IBM zEnterprise 114	CP Assist for Cryptographic Functions Crypto Express3 Coprocessor	Encrypted keys require CEX3C with the Nov. 2009 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	CP Assist for Cryptographic Functions Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	

Related information

You **cannot** overlap the plaintext and ciphertext fields. For example:

```
pppppp
     cccccc  is not supported.

cccccc
     pppppp  is not supported.
```

```
ppppppcccccc is supported.

P represents the plaintext and c represents the ciphertext.
```

Symmetric Key Encipher (CSNBSYE or CSNBSYE1 and CSNESYE or CSNESYE1)

Use the symmetric key encipher callable service to encipher data using one of the supported modes. ICSF supports several processing rules to encipher data. You choose the type of processing rule that the Symmetric Key Encipher callable service should use for the block chaining.

Processing Rule
Purpose

ANSI X9.23
For cipher block chaining. The plaintext may be any length. The ciphertext will be between 1 and 8 bytes longer than the plaintext. This process rule always pads the plaintext during encryption so that ciphertext produced is an exact multiple of the block size, even if the plaintext was already a multiple of the blocksize.

CBC For cipher block chaining. The plaintext must be an exact multiple of the block size for the specified algorithm (8 bytes for DES, 16 bytes for AES). The ciphertext will have the same length as the plaintext.

CBC-CS
For cipher block chaining. The plaintext must be at least the block size for the specified algorithm (8 bytes for DES, 16 bytes for AES). The plaintext will have the same length as the ciphertext.

CFB Performs cipher feedback encryption with the segment size equal to the block size. The plaintext can be of any length. The ciphertext will have the same length as the plaintext.

CFB-LCFB
Performs cipher feedback encryption with the segment size set by the caller. The plaintext can be of any length. The ciphertext will have the same length as the plaintext.

CTR Performs counter mode encryption. The ciphertext can be any length. The plaintext will have the same length as the ciphertext.

CUSP For cipher block chaining. The plaintext can be of any length. The ciphertext will have the same length as the plaintext.

ECB Performs electronic code book encryption. The plaintext must be an exact multiple of the block size for the specified algorithm (8 bytes for DES, 16 bytes for AES). The ciphertext will have the same length as the plaintext.

GCM Perform Galois/Counter mode decryption, which provides both confidentiality and authentication for the plaintext and authentication for the additional authenticated data (AAD). The plaintext can be of any length. The ciphertext will have the same length as the plaintext. Additionally, the authentication tag will be verified before any data is returned.

IPS For cipher block chaining. The plaintext can be of any length. The ciphertext will have the same length as the plaintext.

OFB Perform output feedback mode encryption. The plaintext can be of any length. The ciphertext will have the same length as the plaintext.

PKCS-PAD
For cipher block chaining. The plaintext may be any length. The ciphertext will be between 1 and 8 bytes longer than the plaintext. This process rule always pads the ciphertext so that ciphertext produced is an exact multiple of the blocksize, even if the plaintext was already a multiple of the blocksize.

The Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are supported. AES encryption uses a 128-, 192-, or 256-bit key. DES encryption uses a 56-, 112-, or 168-bit key. See the processing rule descriptions for limitations. For each algorithm, certain processing rules are not allowed. See the rule_array parameter description for more information.

All modes except ECB use an initial chaining vector (ICV) in their processing.

All modes that tolerate chaining produce a resulting chaining value called the output chaining vector (OCV). The application can pass the OCV as the ICV in the next encipher call. This results in record chaining.

The selection between single-DES decryption mode and triple-DES decryption mode is controlled by the length of the key supplied in the *key_identifier* parameter. If a single-length key is supplied, single-DES decryption is performed. If a double-length or triple-length key is supplied, triple-DES decryption is performed.

The key may be specified as a clear key value, an internal clear key token, or the label name of a clear key or an encrypted key in the CKDS.

Choosing between CSNBSYE and CSNBSYE1

CSNBSYE and CSNBSYE1 provide identical functions. When choosing which service to use, consider this:

- **CSNBSYE** requires the cleartext and ciphertext to reside in the caller's primary address space. Also, a program using CSNBSYE adheres to the IBM Common Cryptographic Architecture: Cryptographic Application Programming Interface.

  The callable service name for AMODE(64) invocation is CSNESYE.
- **CSNBSYE1** allows the cleartext and ciphertext to reside either in the caller's primary address space or in a data space. This can allow you to encipher more data with one call. However, a program using CSNBSYE1 does not adhere to the IBM Common Cryptographic Architecture: Cryptographic Application Programming Interface and may need to be modified prior to it running with other cryptographic products that follow this programming interface.

  For CSNBSYE1, *clear_text_id* and *cipher_text_id* are access list entry token (ALET) parameters of the data spaces containing the cleartext and ciphertext.

  The callable service name for AMODE(64) invocation is CSNESYE1.

Format

```
CALL CSNBSYE(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_identifier_length,
             key_identifier,
             key_parms_length,
```

```
                key_parms,
                block_size,
                initialization_vector_length,
                initialization_vector,
                chain_data_length,
                chain_data,
                clear_text_length,
                clear_text,
                cipher_text_length,
                cipher_text,
                optional_data_length,
                optional_data)
CALL CSNBSYE1(
                return_code,
                reason_code,
                exit_data_length,
                exit_data,
                rule_array_count,
                rule_array,
                key_identifier_length,
                key_identifier,
                key_parms_length,
                key_parms,
                block_size,
                initialization_vector_length,
                initialization_vector,
                chain_data_length,
                chain_data,
                clear_text_length,
                clear_text,
                cipher_text_length,
                cipher_text,
                optional_data_length,
                optional_data,
                clear_text_id,
                cipher_text_id)
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Ignored	Integer

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

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	Integer

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

Table 4. Symmetric Key Encipher Rule Array Keywords

Keyword	Meaning
Algorithm (required)	
AES	Specifies that the Advanced Encryption Standard (AES) algorithm is to be used. The block size is 16 bytes. The key length may be 16, 24, or 32 bytes. The *chain_data* field must be at least 32 bytes in length. The OCV is the first 16 bytes in the *chain_data*. AES does not support the CUSP, IPS, or X9.23 processing rules.
DES	Specifies that the Data Encryption Standard (DES) algorithm is to be used. The algorithm, DES or TDES, will be determined from the length of the key supplied. The key length may be 8, 16, or 24. The block size is 8 bytes. The *chain_data* field must be at least 16 bytes in length. The OCV is the first eight bytes in the *chain_data*. DES does not support the CTR or GCM processing rules.
Processing Rule (optional) Rules CBC-CS, CUSP, IPS, PKCS-PAD, and X9.23 should be specified only when there is one request or on the last request of a sequence of chained requests.	
CBC	CBC mode (cipher block chaining). The text length must be a multiple of the block size for the specified algorithm. CBC is the default value.
CBC-CS	CBC mode (cipher block chaining) with ciphertext stealing. The text length must be at least the block size for the specified algorithm.
CFB	CFB mode (cipher feedback) that is compatible with IBM's Encryption Facility product. Input text may be any length.

Table 4. Symmetric Key Encipher Rule Array Keywords (continued)

Keyword	Meaning
CFB-LCFB	CFB mode (cipher feedback). This rule allows the value of s (the segment size) to be something other than the block size (s is set to the block size with the CFB processing rule). The *key_parms_length* and *key_parms* parameters are used to set the value of *s*. Input text may be any length.
CTR	CTR mode (counter mode). Input text may be any length.
CUSP	CBC mode (cipher block chaining) that is compatible with IBM's CUSP and PCF products. Input text may be any length.
ECB	ECB mode (electronic codebook). The text length must be a multiple of the block size for the specified algorithm.
GCM	GCM mode (Galois/Counter Mode). The *key_parms_length* and *key_parms* parameters are used to indicate the length of the tag (the value *t*) on input and contain the tag on output. Additional Authenticated Data (AAD) is contained in the *optional_data_length* and *optional_data* parameters. Input text may be any length.
IPS	CBC mode (cipher block chaining) that is compatible with IBM's IPS product. Input text may be any length.
OFB	OFB mode (output feedback). Input text may be any length.
PKCS-PAD	CBC mode (cipher block chaining) not necessarily in exact multiples of the block length (8 bytes for DES and 16 bytes for AES). PKCS-PAD always pads the plaintext so that the ciphertext produced is an exact multiple of the block length and longer than the plaintext.
X9.23	CBC mode (cipher block chaining) for 1 to 8 bytes of padding added according to ANSI X9.23. Input text may be any length.
Key Rule (optional)	
KEY-CLR	This specifies that the key parameter contains a clear key value. KEY-CLR is the default.
KEYIDENT	This specifies that the *key_identifier* field will be an internal clear token, or the label name of a clear key or encrypted key in the CKDS. Normal CKDS labelname syntax is required.
ICV Selection (optional)	
INITIAL	This specifies taking the initialization vector from the *initialization_vector* parameter. INITIAL is the default value. INITIAL is not valid with processing rule GCM.
CONTINUE	This specifies taking the initialization vector from the output chaining vector contained in the work area to which the *chain_data* parameter points. CONTINUE is not valid for processing rules ECB, GCM, or X9.23.
FINAL	This specifies taking the initialization vector from the output chaining vector contained in the work area to which the *chain_data* parameter points. Using FINAL indicates that this call contains the last portion of data. FINAL is valid for processing rules CBC-CS, CFB, CFB-LCBF, CTR, and OFB.

Table 4. Symmetric Key Encipher Rule Array Keywords (continued)

Keyword	Meaning
ONLY	This specifies taking the initialization vector from the *initialization_vector* parameter and that the entirety of the data to be processed is in this single call. ONLY is valid for processing rules CBC-CS, CFB, CFB-LCFB, CTR, GCM, and OFB.

key_identifier_length

Direction	Type
Input	Integer

The length of the *key_identifier* parameter. For clear keys, the length is in bytes and includes only the value of the key.

For the KEYIDENT keyword, this parameter value must be 64.

key_identifier

Direction	Type
Input	String

For the KEY-CLR keyword, this specifies the cipher key. The parameter must be left justified.

For the KEYIDENT keyword, this specifies an internal clear token, or the label name of a clear key or an encrypted key in the CKDS. Normal CKDS label name syntax is required. The key algorithm may be DES or AES and the key type must be DATA. Encrypted key support is available on IBM System z10 and later servers.

key_parms_length

Direction	Type
Input	Integer

The length of the *key_parms* parameter.

- For the CFB-LCFB and CTR processing rules, this length must be 1.
- For the GCM processing rule, this is the length in bytes of the authentication tag to be generated. Valid lengths are 4, 8, 12, 13, 14, 15, 16. Using a length of 4 or 8 is strongly discouraged.
- For all other processing rules, this field is ignored.

When deciphering the text, you must specify this same length.

key_parms

Direction	Type
Input/Output	String

This parameter contains key-related parameters specific to the encryption algorithm and processing mode.

- For the CFB-LCFB processing rule, this 1-byte field specifies the segment size in bytes. Valid values are 1 to the blocksize, inclusive. The block size is eight for DES and sixteen for AES.
- For the CTR processing rule, this 1-byte field specifies the number of low order bytes of the counter to be incremented. The remaining upper order bytes are the nonce. Valid values are 1 to the block size, inclusive. The blocksize is sixteen for AES.
- For the GCM processing rule, this will contain the generated authentication tag for the provided plaintext (plain_text parameter) and additional authenticated data (optional_data parameter).
- For all other processing rules, this field is ignored.

For the modes where *key_parms* is used, you must specify the same *key_parms* when deciphering the text using the Symmetric Key Decipher callable service.

block_size

Direction	Type
Input	Integer

This parameter contains the processing size of the text block in bytes. This value will be algorithm specific.

initialization_vector_length

Direction	Type
Input	Integer

The length of the *initialization_vector* parameter. This parameter is ignored for the ECB processing rule. For the GCM processing rule, NIST recommends a length of 12, but tolerates any non-zero length. For all other processing rules, the length should be equal to the block length for the algorithm specified.

initialization_vector

Direction	Type
Input	String

This initialization chaining value. You must use the same ICV to decipher the data. This parameter is ignored for the ECB processing rule.

chain_data_length

Direction	Type
Input/Output	Integer

The length of the *chain_data* parameter. On output, the actual length of the chaining vector will be stored in the parameter. This parameter is ignored if the ICV selection keyword in ONLY.

chain_data

Direction	Type
Input/Output	String

This field is used as a system work area for the chaining vector. Your application program must not change the data in this string. The chaining vector holds the output chaining vector from the caller.

The direction is output if the ICV selection keyword is INITIAL. This parameter is ignored if the ICV selection keyword in ONLY.

The mapping of the *chain_data* depends on the algorithm specified. For AES, the *chain_data* field must be at least 32 bytes in length. The OCV is in the first 16 bytes in the *chain_data*. For DES, the *chain_data* field must be at least 16 bytes in length.

clear_text_length

Direction	Type
Input	Integer

The length of the cleartext. A zero value in the *clear_text_length* parameter is not valid except with the GCM processing rule when performing a GMAC operation. The length must be a multiple of the algorithm block size for the CBC, ECB, and PKCS-PAD processing rules, but may be any length with the other processing rules. For the processing rules that support partial blocks (or segments for CFB-LCFB), it is recommended that is the final block (or segment) be the only one that is partial. Having a partial block in the middle is not a supported operation as defined by the standards documents and may not be portable to other encryption systems.

clear_text

Direction	Type
Input	String

The text to be enciphered.

cipher_text_length

Direction	Type
Input/Output	Integer

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

cipher_text

Direction	Type
Output	String

The enciphered text the service returns.

optional_data_length

Direction	Type
Input	Integer

The length of the *optional_data* parameter. For the GCM processing rule, this parameter contains the length of the Additional Authenticated Data (AAD), and may be any length, including zero. For all other processing rules, this field is ignored.

optional_data

Direction	Type
Input	String

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

You must specify the same optional_data when deciphering the text using Symmetric Key Decipher.

clear_text_id

Direction	Type
Input	Integer

For CSNBSYE1 only, the ALET of the clear text to be enciphered.

cipher_text_id

Direction	Type
Input	Integer

For CSNBSYE1 only, the ALET of the ciphertext that the application supplied.

Usage notes

- SAF may be invoked to verify the caller is authorized to use the specified key label stored in the CKDS.
- To use a DES or AES encrypted DATA key in the CKDS, the ICSF segment of the CSFKEYS class general resource profile associated with the specified key label must contain SYMCPACFWRAP(YES).
- No pre- or post-processing exits are enabled for this service.
- The master keys need to be loaded only when using this service with the encrypted key labels.
- The AES algorithm will use hardware if it is available. Otherwise, clear key operations will be performed in software.
- AES has the same availability restrictions as triple-DES.
- This service will fail if execution would cause destructive overlay of the *clear_text* field.

Access control points

When the label of an encrypted key is specified for the *key_identifier* parameter, the appropriate access control point listed below must be enabled.

Table 5. Required access control points for Symmetric Key Encipher

Key algorithm	Access control point
AES	Symmetric Key Encipher/Decipher - Encrypted AES keys
DES	Symmetric Key Encipher/Decipher - Encrypted DES keys

Required hardware

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

Table 6. Symmetric Key Encipher required hardware

Server	Required cryptographic hardware	Restrictions
IBM eServer zSeries 990 IBM eServer zSeries 890	CP Assist for Cryptographic Functions	GCM processing rule is not supported. CFB-LCFB processing rule is supported only when key_parms specifies a segment size equal to the blocksize.
IBM System z9 EC IBM System z9 BC	CP Assist for Cryptographic Functions	GCM processing rule is not supported. CFB-LCFB processing rule is supported only when key_parms specifies a segment size equal to the blocksize.
IBM System z10 EC IBM System z10 BC	CP Assist for Cryptographic Functions Crypto Express3 Coprocessor	GCM processing rule is not supported. CFB-LCFB processing rule is supported only when key_parms specifies a segment size equal to the blocksize. Encrypted keys require the CEX3C with the Nov. 2009 or later licensed internal code (LIC).
IBM zEnterprise 196 IBM zEnterprise 114	CP Assist for Cryptographic Functions Crypto Express3 Coprocessor	Encrypted keys require the CEX3C with the Nov. 2009 or later licensed internal code (LIC).
IBM zEnterprise EC12 IBM zEnterprise BC12	CP Assist for Cryptographic Functions Crypto Express3 Coprocessor Crypto Express4 CCA Coprocessor	

Related information

You **cannot** overlap the plaintext and ciphertext fields. For example:

```
pppppp
     cccccc  is not supported.

cccccc
     pppppp  is not supported.
```

```
ppppppcccccc is supported.

P represents the plaintext and c represents the ciphertext.
```

The method used to produce the OCV is the same with the CBC and X9.23 processing rules. However, that method is different from the method used by the CUSP and IPS processing rules.

PKCS #11 Secret key decrypt (CSFPSKD and CSFPSKD6)

Use the PKCS #11 secret key decrypt callable service to decipher data using a clear symmetric key. AES, DES, BLOWFISH, and RC4 are supported. This service supports CBC, CTR, ECB, Galois/Counter, and stream modes and PKCS #7 padding. The key handle must be a handle of a PKCS #11 secret key object. The CKA_DECRYPT attribute must be true.

If the length of output field is too short to hold the output, the service will fail and return the required length of the output field in the clear_text_length parameter.

The callable service can be invoked in AMODE(24), AMODE(31), or AMODE(64). 64-bit callers must use CSFPSKD6.

Format

```
CALL CSFPSKD(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_handle,
             initialization_vector_length,
             initialization_vector,
             chain_data_length,
             chain_data,
             cipher_text_length,
             cipher_text,
             cipher_text_id,
             clear_text_length,
             clear_text,
             clear_text_id )
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Ignored	Integer

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

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service.

Table 7. Keywords for secret key decrypt

Keyword	Meaning
Encryption Mechanism (Optional. No default. If not specified, mechanism will be taken from key type of secret key. If specified , must match key type)	
AES	AES algorithm will be used.
DES	DES algorithm will be used. This is only single-key encryption.
DES3	DES3 algorithm will be used, This includes double- and triple-key encryption.
BLOWFISH	BLOWFISH algorithm will be used.
RC4	RC4 algorithm will be used. This is a stream cipher.
Processing Rule (optional)	
CBC	Performs cipher block chaining. The cipher text length must be a multiple of the block size for the specified algorithm (8 bytes for DES, DES3, and BLOWFISH, 16 bytes for AES). CBC is the default value for DES, DES3, AES, and BLOWFISH. CBC cannot be specified for RC4.
CBC-PAD	Performs cipher block chaining. The cipher text length must be greater than zero and a multiple of the block size for the specified algorithm. For FINAL and ONLY calls, PKCS #7 padding is performed. For this reason, the clear text will always be shorter than the cipher text and may even be zero length. CBC-PAD cannot be specified for BLOWFISH or RC4.

Table 7. Keywords for secret key decrypt (continued)

Keyword	Meaning
CTR	Performs counter mode decryption. The cipher text length must be greater than zero. The clear text will be the same length as the cipher text. CTR may only be specified for AES.
ECB	Performs electronic code book encryption. The cipher text length must be a multiple of the block size for the specified algorithm. ECB cannot be specified for BLOWFISH or RC4.
GCM	Performs Galois/Counter mode encryption. The cipher text length must be greater than zero. The clear text will be shorter than the cipher text and may even be zero length due to the truncation of the authentication tag. GCM may only be specified with AES. GMAC is a specialized form of GCM where no plain text is returned.
STREAM	Performs a stream cipher. STREAM cannot be specified for BLOWFISH, DES, DES3, or AES. STREAM is the default value for RC4.
Chaining Selection (optional)	
INITIAL	Specifies this is the first call in a series of chained calls. For cipher block chaining, the initialization vector is taken from the *initialization_vector* parameter. Cannot be specified with processing rule ECB or GCM.
CONTINUE	Specifies this is a middle call in a series of chained calls. Intermediate results are read from and stored in the *chain_data* field. Cannot be specified with processing rule ECB or GCM.
FINAL	Specifies this is the last call in a series of chained calls. Intermediate results are read from the *chain_data* field. Cannot be specified with processing rule ECB or GCM.
ONLY	Specifies this is the only call and the call is not chained. For cipher block chaining, the initialization vector is taken from the *initialization_vector* parameter. For counter mode and Galois Counter mode, the initialization parameters are taken from the *initialization_vector* parameter. ONLY is the default chaining.

key_handle

Direction	Type
Input	String

The 44-byte handle of secret key object.

initialization_vector_length

Direction	Type
Input	Integer

Length of the *initialization_vector* in bytes. For CBC and CBC-PAD, this must be 8 bytes for DES and BLOWFISH and 16 bytes for AES. For CTR, this must be the size of the structure described in the *initialization_vector* parameter (17 bytes). For GCM, this must be the size of the *initialization_vector* field (28 bytes).

initialization_vector

Direction	Type
Input	String

This field has a varying format depending on the mechanism specified. For CBC and CBC-PAD this is the 8 or 16 byte initial chaining value. The formats for GCM and CTR are shown in the following tables.

Table 8. initialization_vector parameter format for GCM mechanism

Offset	Length in bytes	Direction	Description
0	4	Input	Length in bytes of the initialization vector. The minimum value is 1. The maximum value is 128. 12 is recommended.
4	8	Input	64-bit address of the initialization vector. The data must reside in the caller's address space. High order word must be set to all zeros by AMODE31 callers.
12	4	Input	Length in bytes of the additional authentication data. The minimum value is 0. The maximum value is 1048576.
16	8	Input	64-bit address of the additional authentication data. The data must reside in the caller's address space. High order word must be set to all zeros by AMODE31 callers. This field is ignored if the length of the additional authentication data is zero.
24	4	Input	Length in bytes of the desired authentication tag. This value must be one of 4, 8, 12, 13, 14, 15, or 16.

Table 9. initialization_vector parameter format for CTR mechanism

Offset	Length in bytes	Direction	Description
0	16	Input	Initial counter block.
16	1	Input	The number of low order bytes of the counter to be incremented. The remaining upper order bytes are the nonce. Valid values are 1 to the block size, inclusive. The block size is sixteen for AES.

chain_data_length

Direction	Type
Input/Output	Integer

The byte length of the chain_data parameter. This must be 128 bytes.

chain_data

Direction	Type
Input/Output	String

This field is a 128-byte work area. The chain data permits chaining data from one call to another. ICSF initializes the chain data on an INITIAL call, and may change it on subsequent CONTINUE calls. Your application must not change the data in this field between the sequence of INITIAL, CONTINUE, and FINAL calls for a specific message. The chain data has the following format:

Table 10. chain_data parameter format

Offset	Length	Description	
0	4	Flag word	
		Bit	**Meaning when set on**
		0	Cryptographic state object has been allocated
		1-31	Reserved for IBM's use

Table 10. chain_data parameter format (continued)

Offset	Length	Description
4	44	Cryptographic state object handle
48	80	Reserved for IBM's use

cipher_text_length

Direction	Type
Input	Integer

Length of the *cipher_text* parameter in bytes. Except for processing rule GCM, the length can be up to 2147483647. For processing rule GCM, the length cannot exceed 1048576 plus the length of the tag.

cipher_text

Direction	Type
Input	String

Text to be decrypted.

cipher_text_id

Direction	Type
Input	Integer

The ALET identifying the space where the cipher text resides.

clear_text_length

Direction	Type
Input/Output	Integer

On input, the length in bytes of the *clear_text* parameter. On output, the length of the text decrypted into the *clear_text* parameter

clear_text

Direction	Type
Output	String

Decrypted text

clear_text_id

Direction	Type
Input	Integer

The ALET identifying the space where the clear text resides.

Authorization

To use this service with a public object, the caller must have at least SO (READ) authority or USER (READ) authority (any access).

To use this service with a private object, the caller must have at least USER (READ) authority (user access).

Usage Notes

If the INITIAL rule is used to start a series of chained calls:

- The same key_handle, Encryption Mechanism and Processing Rule must be used on the subsequent CONTINUE and FINAL calls.
- The key used to initiate the chained calls must not be deleted until the chained calls are complete.
- The application should make a FINAL call to free ICSF resources allocated. If processing is to be aborted without making a FINAL call and the *chain_data* parameter indicates that a cryptographic state object has been allocated, the caller must free the object by calling CSFPTRD (or CSFPTRD6 for 64-bit callers) passing the state object's handle.

GCM decryption may be used to verify a GMAC on some authentication data. To do this request AES decryption with processing rule. The *cipher_text_length* and *cipher_text* fields must be set to the length and value of the GMAC to be verified. A *return_code* of zero and no *clear_text* data returned means the GMAC verification was successful.

A secure key may not be used for Processing Rules CTR or GCM.

PKCS #11 Secret key encrypt (CSFPSKE and CSFPSKE6)

Use the PKCS #11 secret key encrypt callable service to encipher data using a clear symmetric key. AES, DES, BLOWFISH, and RC4 are supported. This service supports CBC, CTR, ECB, Galois/Counter, and stream modes and PKCS #7 padding. The key handle must be a handle of a PKCS #11 secret key object. The CKA_ENCRYPT attribute must be true.

If the length of output field is too short to hold the output, the service will fail and return the required length of the output field in the *cipher_text_length* parameter.

The callable service can be invoked in AMODE(24), AMODE(31), or AMODE(64). 64-bit callers must use CSFPSKE6.

Format

```
CALL CSFPSKE(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_handle,
             initialization_vector_length,
             initialization_vector,
             chain_data_length,
             chain_data,
             clear_text_length,
             clear_text,
             clear_text_id,
             cipher_text_length,
             cipher_text,
             cipher_text_id )
```

Parameters

return_code

Direction	Type
Output	Integer

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

reason_code

Direction	Type
Output	Integer

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

exit_data_length

Direction	Type
Ignored	Integer

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

exit_data

Direction	Type
Ignored	String

This field is ignored.

rule_array_count

Direction	Type
Input	Integer

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

rule_array

Direction	Type
Input	String

Keywords that provide control information to the callable service.

Table 11. Keywords for secret key encrypt

Keyword	Meaning
Encryption Mechanism (Optional. No default. If not specified, mechanism will be taken from key type of secret key. If specified , must match key type)	
AES	AES algorithm will be used.
DES	DES algorithm will be used. This is only single-key encryption.
DES3	DES3 algorithm will be used, This includes double- and triple-key encryption.

Table 11. Keywords for secret key encrypt (continued)

Keyword	Meaning
BLOWFISH	BLOWFISH algorithm will be used.
RC4	RC4 algorithm will be used. This is a stream cipher.
Processing Rule (optional)	
CBC	Performs cipher block chaining. The text length must be a multiple of the block size for the specified algorithm (8 bytes for DES, DES3, and BLOWFISH, 16 bytes for AES). CBC is the default value for DES, DES3, AES, and BLOWFISH. CBC cannot be specified for RC4.
CBC-PAD	Performs cipher block chaining. Except for FINAL and ONLY chaining calls, the clear text length must be a multiple of the block size for the specified algorithm. For FINAL and ONLY calls: • The clear text length may be shorter than the block size and may even be zero. • PKCS #7 padding is performed. Thus, the cipher text will always be longer than the clear text. CBC-PAD cannot be specified for BLOWFISH or RC4.
CTR	Performs counter mode encryption. The clear text length must be greater than zero. The cipher text will be the same length as the clear text. CTR may only be specified for AES.
ECB	Performs electronic code book encryption. The text length must be a multiple of the block size for the specified algorithm. ECB cannot be specified for BLOWFISH or RC4.
GCM	Performs Galois/Counter mode encryption. The clear text length may be shorter than the block size and may even be zero. The authentication tag is returned appended to the cipher text. GCM may only be specified with AES. GMAC is a specialized form of GCM where no plain text is specified.
GCMIVGEN	Performs similarly to the GCM processing rule except that ICSF will generate part of the initialization vector and return it in the *initialization_vector* parameter. Having ICSF generate the initialization vector ensures that initialization vectors are never repeated for a given key object.
STREAM	Performs a stream cipher. STREAM cannot be specified for BLOWFISH, DES, DES3, or AES. STREAM is the default value for RC4.
Chaining Selection (optional)	
INITIAL	Specifies this is the first call in a series of chained calls. For cipher block chaining, the initialization vector is taken from the *initialization_vector* parameter. Intermediate results are stored in the chain_data field. Cannot be specified with processing rule ECB, GCM, or GCMIVGEN.
CONTINUE	Specifies this is a middle call in a series of chained calls. Intermediate results are read from and stored in the *chain_data* field. Cannot be specified with processing rule ECB, GCM, or GCMIVGEN.
FINAL	Specifies this is the last call in a series of chained calls. Intermediate results are read from the *chain_data* field. Cannot be specified with processing rule ECB, GCM, or GCMIVGEN.

Table 11. Keywords for secret key encrypt (continued)

Keyword	Meaning
ONLY	Specifies this is the only call and the call is not chained. For cipher block chaining, the initialization vector is taken from the *initialization_vector* parameter. For counter mode and Galois Counter mode, the initialization parameters are taken from the *initialization_vector* parameter. ONLY is the default chaining.

key_handle

Direction	Type
Input	String

The 44-byte handle of secret key object.

initialization_vector_length

Direction	Type
Input	Integer

Length of the *initialization_vector* in bytes. For CBC and CBC-PAD, this must be 8 bytes for DES and BLOWFISH and 16 bytes for AES. For CTR, this must be the size of the structure described in the *initialization_vector* parameter (17 bytes). For GCM and GCMVGEN, this must be the size of the *initialization_vector* field (28 bytes).

initialization_vector

Direction	Type
Input	String

This field has a varying format depending on the mechanism specified. For CBC and CBC-PAD this is the 8 or 16 byte initial chaining value. The formats for GCM, GCMIVGEN, and CTR are shown in the following tables.

Table 12. initialization_vector parameter format for GCM mechanism

Offset	Length in bytes	Direction	Description
0	4	Input	length in bytes of the initialization vector area. The minimum value is 1. The maximum value is 128. 12 is recommended.
4	8	Input	64-bit address of the initialization vector area. The data must reside in the caller's address space. High order word must be set to all zeros by AMODE31 callers.
12	4	Input	length in bytes of the additional authentication data. The minimum value is 0. The maximum value is 1048576.
16	8	Input	64-bit address of the additional authentication data. The data must reside in the caller's address space. High order word must be set to all zeros by AMODE31 callers. This field is ignored if the length of the additional authentication data is zero.
24	4	Input	Length in bytes of the desired authentication tag. This value must be one of 4, 8, 12, 13, 14, 15, or 16.

Table 13. initialization_vector parameter format for GCMIVGEN mechanism

Offset	Length in bytes	Direction	Description
0	4	Input	Nonce value which ICSF is to use as the first 4 bytes of the initialization vector. The remaining 8 bytes will be generated and returned to the caller in the initialization vector area.
4	8	Input	64-bit address of the initialization vector area into which ICSF will store the 8 bytes it generates. The area must reside in the caller's address space. High order word must be set to all zeros by AMODE31 callers. The complete initialization vector to be used for decryption is the 4-byte nonce concatenated with the 8 bytes stored in the area
12	4	Input	length in bytes of the additional authentication data. The minimum value is 0. The maximum value is 1048576.
16	8	Input	64-bit address of the additional authentication data. The data must reside in the caller's address space. High order word must be set to all zeros by AMODE31 callers. This field is ignored if the length of the additional authentication data is zero.
24	4	Input	Length in bytes of the desired authentication tag. This value must be one of 4, 8, 12, 13, 14, 15, or 16.

Table 14. initialization_vector parameter format for CTR mechanism

Offset	Length in bytes	Direction	Description
0	16	Input	Initial counter block.
16	1	Input	The number of low order bytes of the counter to be incremented. The remaining upper order bytes are the nonce. Valid values are 1 to the block size, inclusive. The block size is sixteen for AES.

chain_data_length

Direction	Type
Input/Output	Integer

The byte length of the *chain_data* parameter. This must be 128 bytes.

chain_data

Direction	Type
Input/Output	String

This field is a 128-byte work area. The chain data permits chaining data from one call to another. ICSF initializes the chain data on an INITIAL call, and may change it on subsequent CONTINUE calls. Your application must not change the data in this field between the sequence of INITIAL, CONTINUE, and FINAL calls for a specific message. The chain data has the following format:

Table 15. chain_data parameter format

Offset	Length	Description
0	4	Flag word **Bit** **Meaning when set on** **0** Cryptographic state object has been allocated **1-31** Reserved for IBM's use

Table 15. chain_data parameter format (continued)

Offset	Length	Description
4	44	Cryptographic state object handle
48	80	Reserved for IBM's use

clear_text_length

Direction	Type
Input	Integer

Length of the *clear_text* parameter in bytes. Except for processing rules GCM and GCMIVGEN, the length can be up to 2147483647. For processing rules GCM and GCMIVGEN, the length cannot exceed 1048576.

clear_text

Direction	Type
Input	String

Text to be encrypted

clear_text_id

Direction	Type
Input	Integer

The ALET identifying the space where the clear text resides.

cipher_text_length

Direction	Type
Input/Output	Integer

On input, the length in bytes of the *cipher_text* parameter. On output, the length of the text encrypted into the *cipher_text* parameter.

cipher_text

Direction	Type
Output	String

Encrypted text

cipher_text_id

Direction	Type
Output	Integer

The ALET identifying the space where the cipher text resides.

Authorization

To use this service with a public object, the caller must have at least SO (READ) authority or USER (READ) authority (any access).

To use this service with a private object, the caller must have at least USER (READ) authority (user access).

Usage Notes

If the INITIAL rule is used to start a series of chained calls:

- The same key_handle, Encryption Mechanism and Processing Rule must be used on the subsequent CONTINUE and FINAL calls.
- The key used to initiate the chained calls must not be deleted until the chained calls are complete.
- The application should make a FINAL call to free ICSF resources allocated. If processing is to be aborted without making a FINAL call and the *chain_data* parameter indicates that a cryptographic state object has been allocated, the caller must free the object by calling CSFPTRD (or CSFPTRD6 for 64-bit callers) passing the state object's handle.

GCM encryption may be used to produce a GMAC on some authentication data. To do this, request AES encryption with processing rule GCM or GCMVGEN. The *clear_text_length* field must be set to zero. The authentication tag (the GMAC) is returned in the *cipher_text* field.

For Processing Rule GCMIVGEN, the total number of initialization vector generations for a token key object is limited to 4294967295. Once this number is exceeded, the key object will no longer be eligible for Processing Rule GCMIVGEN and is considered "retired". This usage counter is maintained in the TKDS as part of the key object. For keys that are copied using CSFPTRC (C_CopyObject), the existing counter value is copied to the new key object, but not synchronized after that.

For Processing Rule GCMIVGEN, session key objects have no maximum lifetime. They may be retired at any time. Once retired, the key object will no longer be eligible for Processing Rule GCMIVGEN.

For Processing Rule GCMIVGEN, the nonce value portion of the initialization vector is predetermined by the caller. It is used to ensure that initialization vector values are not repeated for any given key value. The caller should provide a random value and change the value as often as practical. It must be changed whenever:

- A given key value is replicated as a new persistent key object
- A given persistent key object is replicated as a new session key object
- A given session key value is re-instantiated after system IPL
- A given key value is re-instantiated after ICSF indicates it has been retired

Use of Processing Rule GCMIVGEN with token key objects requires that the first 4 bytes of ECVTSPLX or CVTSNAME be set to a unique value with respect to other systems. See *z/OS Cryptographic Services ICSF System Programmer's Guide* for information on how to set these fields.

A session key object should never be used for Processing Rule GCMIVGEN if the key value is distributed to multiple systems outside the current sysplex where new initialization vectors may be generated. Use only token key objects in such cases. If session key objects are used, the other systems must use different nonces.

For Processing Rule GCMIVGEN, the 8 bytes of generated initialization vector are stored back into the initialization vector area before the GCM operation is performed. This allows the generated initialization vector to be part of the additional authentication data, if desired.

| A secure key may not be used for Processing Rules CTR or GCM.

Printed in USA