ICSF

CFB Mode SPE OA19177

Contents

Chapter 1. Symmetric Key Decipher (CSNBSYD and CSNBSYD1)

Use the symmetric key decipher callable service to decipher data in an address space or a data space using the cipher block chaining, electronic code book or cipher feedback modes. ICSF supports the following processing rules to decipher data. You choose the type of processing rule that the 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 8 bytes, but the plaintext will be 1 to 8 bytes shorter than the ciphertext.
CBC	For cipher block chaining. The ciphertext must be an exact multiple of 8 bytes, and the plaintext will have the same length.
CFB	Performs cipher feedback encryption. The ciphertext can be of any length. The plaintext will have the same length as the ciphertext.
CUSP	For cipher block chaining, but the ciphertext can be of any length. The plaintext will be the same length as the ciphertext.
ECB	Performs electronic code book encryption. The text length must be a multiple of the block size for the specified algorithm.
IPS	For cipher block chaining, but the ciphertext can be of any length. The plaintext will be the same length as the ciphertext.

The Advanced Encryption Standard (AES) and DES (Data Encryption Standard) are supported. AES encryption uses a 128-, 192-, or 256-bit key. The CBC, ECB, and CFB modes are supported. Due to export regulations, AES encryption may not be available on your system.

This service supports electronic code book (ECB), cipher block chaining (CBC), cipher feedback (CFB) modes. The CBC and CFB modes of operation use an initial chaining vector (ICV) in their processing. During CBC mode processing, the ICV is exclusive ORed with the first block of plaintext after the decryption step, and thereafter, each block of ciphertext is exclusive ORed with the next block of plaintext after decryption, and so on. For CFB mode processing, the ICV is first encrypted, then exclusive ORed with the first block of ciphertext, and thereafter, the block of exclusive ORed data is encrypted then exclusive ORed with the next block of ciphertext, and so on.

Both Cipher block chaining and Cipher feedback mode also 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 electronic code book mode does not use the initial chaining vector.

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 or the *key_identifier* of a clear key token or labelname in the CKDS.

The callable service Symmetric Key Decipher (without ALET) supports invocation in AMODE(64). The callable service name for AMODE(64) invocation is CSNESYD.

Choosing Between CSNBSYD and CSNBSYD1

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

- **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.
- **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 before it can run 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.

Format

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

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

Parameters

return_code

Direction: Output Type: Integer

The return code specifies the general result of the callable service. Appendix A, ICSF and TSS Return and Reason Codes in *z/OS Cryptographic Services ICSF Application Programmer's Guide* lists the return codes.

reason_code

Direction: Output Type: Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, ICSF and TSS Return and Reason Codes in *z/OS Cryptographic Services ICSF Application Programmer's Guide* lists the reason codes.

exit_data_length

Direction: Ignored Type: Integer

Reserved field.

exit_data

Direction: Ignored Type: String

Reserved field.

rule_array_count

Direction: Input Type: Integer

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

rule_array

Direction: Input Type: 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*. The supported processing rules for AES are CBC, ECB, and CFB.
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*. The processing rules supported for DES are CBC, ECB, X9.23, CUSP, IPS, and CFB.
Processing Rule (optional)	
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.
CFB	CFB mode (cipher feedback) that is compatible with IBM's Encryption Facility product. 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.
IPS	CBC mode (cipher block chaining) that is compatible with IBM's IPS product. Input text may be any length.
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 labelname of a key in the CKDS. Normal CKDS labelname syntax is required. Valid with DES and AES.
ICV Selection (optional)	

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

Keyword	Meaning
INITIAL	This specifies taking the initialization vector from the *initialization_vector* parameter. INITIAL is the default value.
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 valid for processing rules CBC, CFB, IPS, and CUSP only.

key_length

Direction: Input Type: Integer

The length of the key 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: Input Type: 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 labelname of a key in the CKDS. Normal CKDS labelname syntax is required. KEYIDENT is valid with DES and AES.

key_parms_length

Direction: Ignored Type: Integer

The length of the *key_parms* parameter. The maximum size is 256 bytes.

key_parms

Direction: Ignored Type: String

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

block_size

Direction: Input Type: 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: Input Type: Integer

The length of the *initialization_vector* parameter. The length should be equal to the block length for the algorithm specified.

initialization_vector

Direction: Input Type: String

This initialization chaining value for CBC encryption. You must use the same ICV that was used to encipher the data.

chain_data_length

Direction: Input/Output Type: Integer

The length of the *chain_data* parameter. On output, the actual length of the chaining vector will be stored in the parameter.

chain_data

Direction: Input/Output Type: 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.

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: Input Type: Integer

The length of the cipher text. A zero value in the *clear_text_length* parameter is not valid. The length must be a multiple of the algorithm block size.

| Ciphertext may be any length with the CFB, CUSP, or IPS keywords.

cipher_text

Direction: Input Type: String

The text to be deciphered.

clear_text_length

Direction: Input/Output Type: 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.

Input text may be any length with the CUSP keyword.

clear_text

Direction: Output Type: String

The deciphered text the service returns.

optional_data_length

Direction: Ignored Type: Integer

The length of the *optional_data* parameter.

optional_data

Direction: Ignored Type: String

Optional data required by a specified algorithm.

cipher_text_id

Direction: Input Type: Integer

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

clear_text_id

Direction: Input Type: Integer

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

Usage Notes

- No pre- or post-processing exits are enabled for this service.
- No SAF authorization check is made.
- The master keys need not be loaded to use this service.
- The AES algorithm is implemented in software. If available for ECB and CBC mode on HCR7730 and above, hardware will be used.

  If available for CFB mode on HCR7720 and above, hardware will be used.
- AES has the same availability restrictions as triple-DES.
- This service will fail if execution would cause destructive overlay of the *cipher_text* field.

Table 2. Symmetric Key Decipher required hardware

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900	Cryptographic Coprocessor Feature	DES keyword is not supported.
IBM @server zSeries 990 IBM @server zSeries 890	CP Assist for Cryptographic Functions	
IBM System z9 EC and z9 BC		

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.
```

Appendix F, Cryptographic Algorithms and Processes in *z/OS Cryptographic Services ICSF Application Programmer's Guide* discusses the cipher processing rules.

Chapter 2. Symmetric Key Encipher (CSNBSYE and CSNBSYE1)

Use the symmetric key encipher callable service to encipher data in an address space or a data space using the cipher block chaining, electronic code book, or cipher feedback modes. ICSF supports the following processing rules to encipher data. You choose the type of processing rule that the encipher callable service should use for the block chaining.

Processing Rule	Purpose
ANSI X9.23	For block chaining not necessarily in exact multiples of 8 bytes. This process rule pads the plaintext so that ciphertext produced is an exact multiple of 8 bytes.
CBC	For block chaining in exact multiples of 8 bytes.
CFB	Performs cipher feedback encryption. The plaintext can be of any length. The ciphertext will have the same length as the plaintext.
CUSP	For block chaining not necessarily in exact multiples of 8 bytes. The ciphertext will be the same length as the plaintext.
ECB	Performs electronic code book encryption. The text length must be a multiple of the block size for the specified algorithm.
IPS	For block chaining not necessarily in exact multiples of 8 bytes. The ciphertext will be the same length as the plaintext.

The Advanced Encryption Standard (AES) and DES (Data Encryption Standard) are supported. AES encryption uses a 128-, 192-, or 256-bit key. The CBC, CFB, and ECB modes are supported. Due to export regulations, AES encryption may not be available on your system.

This service supports electronic code book (ECB), cipher block chaining (CBC), and cipher feedback (CFB) modes. The CBC and CFB modes of operation use an initial chaining vector (ICV) in their processing. During CBC mode processing, the ICV is exclusive ORed with the first block of plaintext before the encryption step, and thereafter, the block of ciphertext just produced is exclusive ORed with the next block of plaintext, and so on. This disguises any pattern that may exist in the plaintext. For CFB mode processing, the ICV is first encrypted, then exclusive ORed with the first block of plaintext, and thereafter, the block of exclusive ORed data is encrypted then exclusive ORed with the next block of plaintext, and so on. This disguises any pattern that may exist in the plaintext.

Both Cipher block chaining and Cipher Feedback mode also 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 electronic code book mode does not use the initial chaining vector.

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 or the *key_identifier* of a clear key token or labelname in the CKDS.

The callable service Symmetric Key Decipher (without ALET) supports invocation in AMODE(64). The callable service name for AMODE(64) invocation is CSNESYE.

Choosing between CSNBSYE and CSNBSYE1

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

- **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.
- **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 before it can run 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.

Format

```
CALL CSNBSYE(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_length,
             key_identifier,
             key_parms_length,
             key_parms,
             block_size,
             initialization_vector_length,
             initialization_vector,
             chain_data_length,
             chain_data,
             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_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: Output Type: Integer

The return code specifies the general result of the callable service. Appendix A, ICSF and TSS Return and Reason Codes in *z/OS Cryptographic Services ICSF Application Programmer's Guide* lists the return codes.

reason_code

Direction: Output Type: Integer

The reason code specifies the result of the callable service that is returned to the application program. Each return code has different reason codes assigned to it that indicate specific processing problems. Appendix A, ICSF and TSS Return and Reason Codes in *z/OS Cryptographic Services ICSF Application Programmer's Guide* lists the reason codes.

exit_data_length

Direction: Ignored Type: Integer

Reserved field.

exit_data

Direction: Ignored Type: String

Reserved field.

rule_array_count

Direction: Input Type: Integer

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

rule_array

Direction: Input Type: 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 3. Symmetric Key Encipher Rule Array Keywords

Keyword	Meaning
Algorithm (required)	
AES	Specifies that the Advanced Encryption Standard (AES) algorithm is to be used. On systems that contain a Cryptographic Coprocessor Feature, AES is the only algorithm that is supported. 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*. The supported processing rules for AES are CBC, ECB, and CFB.
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*. The processing rules supported for DES are CBC, ECB, X9.23, CUSP, IPS, and CFB.
Processing Rule (optional)	
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.
CFB	CFB mode (cipher feedback) that is compatible with IBM's Encryption Facility product. 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.
IPS	CBC mode (cipher block chaining) that is compatible with IBM's IPS product. Input text may be any length.
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 labelname of a key in the CKDS. Normal CKDS labelname syntax is required. Valid with DES and AES.

Table 3. Symmetric Key Encipher Rule Array Keywords (continued)

Keyword	Meaning
ICV Selection (optional)	
INITIAL	This specifies taking the initialization vector from the *initialization_vector* parameter. INITIAL is the default value.
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 valid for processing rules CBC, CFB, IPS, and CUSP only.

key_length

Direction: Input Type: Integer

The length of the key 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: Input Type: String

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

For the KEYIDENT keyword, this specifies a internal clear token or the labelname of a key in the CKDS. Normal CKDS labelname syntax is required. KEYIDENT is valid with DES and AES.

key_parms_length

Direction: Ignored Type: Integer

The length of the *key_parms* parameter.

key_parms

Direction: Ignored Type: String

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

block_size

Direction: Input Type: Integer

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

initialization_vector_length

Direction: Input Type: Integer

The length of the *initialization_vector* parameter. The length should be equal to the block length for the algorithm specified.

initialization_vector

Direction: Input Type: String

This initialization chaining value for CBC encryption. You must use the same ICV to decipher the data.

chain_data_length

Direction: Input/Output Type: Integer

The length of the *chain_data* parameter. On output, the actual length of the chaining vector will be stored in the parameter.

chain_data

Direction: Input/Output Type: 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.

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: Input Type: Integer

The length of the clear text. A zero value in the *clear_text_length* parameter is not valid. The length must be a multiple of the algorithm block size.

| Input text may be any length with the CFB, CUSP, or IPS keywords.

clear_text

Direction: Input Type: String

The text to be enciphered.

cipher_text_length

Direction: Input/Output Type: 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: Output Type: String

The enciphered text the service returns.

optional_data_length

Direction: Ignored Type: Integer

The length of the *optional_data* parameter.

optional_data

Direction: Ignored Type: String

Optional data required by a specified algorithm.

clear_text_id

Direction: Input Type: Integer

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

cipher_text_id

Direction: Input Type: Integer

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

Usage Notes

- No pre- or post-processing exits are enabled for this service.
- No SAF authorization check is made.
- The master keys need not be loaded to use this service.
- The AES algorithm is implemented in software. If available for ECB and CBC mode on HCR7730 and above, hardware will be used.

  If available for CFB mode on HCR7720 and above, hardware will be used.
- AES has the same availability restrictions as triple-DES.
- This service will fail if execution would cause destructive overlay of the *clear_text* field.

Table 4. Symmetric Key Encipher required hardware

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900	Cryptographic Coprocessor Feature	DES keyword is not supported.
IBM @server zSeries 990 IBM @server zSeries 890	CP Assist for Cryptographic Functions	
IBM System z9 EC and z9 BC		

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.

Appendix F, Cryptographic Algorithms and Processes in *z/OS Cryptographic Services ICSF Application Programmer's Guide* discusses the cipher processing rules.