z/OSCryptographic Services
Integrated Cryptographic Service Facility

PKCS#1 OAEP data block formatting for symmetric key wrapping using the SHA-256 hash method (APAR OA36705)

September, 2011

Contents

Chapter 1. Overview

The PTF for APAR OA36705 can be applied to ICSF HCR7780 to extend ICSF's support of PKA RSA PKCS#1 Optimal Asymmetric Encryption Padding (OAEP). Without this PTF, ICSF HCR7780 supports symmetric key export, import and generation using PKCS#1 OAEP data block formatting with the SHA-1 hash method. By applying PTF for APAR OA36705, ICSF HCR7780 will also support the SHA-256 hash method.

Specifically, the following callable services are updated:

- Symmetric Key Export (CSNDSYX and CSNFSYX)
- Symmetric Key Generate (CSNDSYG and CSNFSYG)
- Symmetric Key Import (CSNDSYI and CSNFSYI)

This document contains alterations to the information previously presented in the manual *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SA22-7522-14.

Alterations to the information are identified in this document by revision bars (|) in the left margin of the page.

Chapter 2. Update of z/OS Cryptographic Services ICSF Application Programmer's Guide, SA22-7522-14, information

This chapter contains updates to the document *z/OS Cryptographic Services ICSF Application Programmer's Guide*, SA22-7522-14. Refer to this source document if background information is needed.

Symmetric Key Export (CSNDSYX and CSNFSYX)

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

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

Format

```
CALL CSNDSYX(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             source_key_identifier_length,
             source_key_identifier,
             RSA_public_key_identifier_length,
             RSA_public_key_identifier,
             RSA_enciphered_key_length,
             RSA_enciphered_key)
```

Parameters

return_code

Direction: Output Type: Integer

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

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.

exit_data_length

Direction: Input/Output Type: Integer

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

exit_data

Direction: Input/Output Type: String

The data that is passed to the installation exit.

rule_array_count

Direction: Input Type: Integer

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

rule_array

Direction: Input Type: String

Keywords that provide control information to the callable service. Table 1 lists the keywords. Each keyword is left-justified in 8-byte fields and padded on the right with blanks. All keywords must be in contiguous storage.

For HMAC keys, only the PKOAEP2 recovery method is supported.

Table 1. Keywords for Symmetric Key Export Control Information

Keyword	Meaning
Algorithm (One keyword, optional)	
AES	The key being exported is an AES key.
DES	The key being exported is a DES key. This is the default.
HMAC	The key being exported is an HMAC key. Only the PKOAEP2 key formatting method is supported.
Key Formatting method (One required)	
PKCSOAEP	Specifies to format the key according to the method in RSA DSI PKCS #1V2 OAEP. The default hash method is SHA-1. Use the SHA-256 keyword for the SHA-256 hash method.
PKCS–1.2	Specifies to format the key according the method found in RSA DSI PKCS #1 block type 02 to recover the symmetric key.
PKOAEP2	Specifies to format the key according to the method found in RSA DSI PKCS #1 v2.1 RSAES-OAEP documentation. Only valid with HMAC algorithm. A hash method is required.
ZERO-PAD	The clear key is right-justified in the field provided, and the field is padded to the left with zeros up to the size of the RSA encryption block (which is the modulus length).
Hash Method (One, optional for PKCSOAEP, required for PKOAEP2. Not valid with any other Key Formatting method)	
SHA-1	Specifies to use the SHA-1 hash method to calculate the OAEP message hash. This is the default for PKCSOAEP.
SHA-256	Specifies to use the SHA-256 hash method to calculate the OAEP message hash.

Table 1. Keywords for Symmetric Key Export Control Information (continued)

Keyword	Meaning
SHA-384	Specifies to use the SHA-384 hash method to calculate the OAEP message hash. Not valid with PKCSOAEP.
SHA-512	Specifies to use the SHA-512 hash method to calculate the OAEP message hash. Not valid with PKCSOAEP.

source_key_identifier_length

Direction: Input Type: Integer

The length of the *source_key_identifier* parameter. The minimum size is 64 bytes. The maximum size is 725 bytes.

source_key_identifier

Direction: Input/Output Type: String

The label or internal token of a secure AES DATA, DES DATA, or HMAC key to encrypt under the supplied RSA public key. The key in the key identifier must match the algorithm in the *rule_array*. DES is the default algorithm.

RSA_public_key_identifier_length

Direction: Input Type: Integer

The length of the *RSA_public_key_identifier* parameter. The maximum size is 3500 bytes.

RSA_public_key_identifier

Direction: Input Type: String

A PKA public key token or label of the key to protect the exported symmetric key.

RSA_enciphered_key_length

Direction: Input/Output Type: Integer

The length of the *RSA_enciphered_key* parameter. This is updated with the actual length of the *RSA_enciphered_key* generated. The maximum size you can specify in this parameter is 512 bytes, although the actual key length may be further restricted by your hardware configuration (as shown in Table 3 on page 7).

RSA_enciphered_key

Direction: Output Type: String

This field contains the RSA enciphered key, protected by the public key specified in the *RSA_public_key_identifier* field.

Restrictions

If you are running with the Cryptographic Coprocessor Feature, the enhanced system keys must be present in the CKDS.

Usage Notes

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

The hardware configuration sets the limit on the modulus size of keys for key management; thus, this service will fail if the RSA key modulus bit length exceeds this limit.

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

Table 2. Required access control points for Symmetric Key Export

Key formatting method	**Algorithm**	**Access control point**
PKCSOAEP	AES	Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2
	DES	Symmetric Key Export - DES, PKCS-1.2
PKCS-1.2	AES	Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2
	DES	Symmetric Key Export - DES, PKCS-1.2
ZERO-PAD	AES	Symmetric Key Export - AES, ZERO-PAD
	DES	Symmetric Key Export - DES, ZERO-PAD
PKOAEP2	HMAC	Symmetric Key Export - HMAC, PKOAEP2

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

Table 3. Symmetric key export required hardware

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900	Cryptographic Coprocessor Feature	RSA keys with moduli greater than 1024-bit length are not supported. Encrypted AES keys are not supported. The DES, HMAC, and PKOAEP2 keywords are not supported.
	PCI Cryptographic Coprocessor	ICSF routes this service to a PCI Cryptographic Coprocessor if one is available on your server. This service will not be routed to a PCI Cryptographic Coprocessor if the modulus bit length of the RSA public key is less than 512 bits. Use of keyword PKCSOAEP requires the PCI Cryptographic Coprocessor and uses the SHA-1 hash method. The SHA-256 keyword is not supported for PKCSOAEP. RSA keys with moduli greater than 2048-bit length are not supported. Encrypted AES keys are not supported. The DES, HMAC, and PKOAEP2 keywords are not supported.
IBM @server zSeries 990 IBM @server zSeries 890	PCI X Cryptographic Coprocessor Crypto Express2 Coprocessor	RSA keys with moduli greater than 2048-bit length are not supported. Encrypted AES keys are not supported. The HMAC and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP.
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Encrypted AES keys require the Nov. 2008 or later licensed internal code (LIC). The HMAC and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP.

Table 3. Symmetric key export required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Encrypted AES keys require the Nov. 2008 or later licensed internal code (LIC). The HMAC and PKOAEP2 keywords are not supported. The SHA-256 keyword is not supported for PKCSOAEP.
	Crypto Express3 Coprocessor	HMAC keys not supported. The SHA-256 keyword is not supported for PKCSOAEP.
z196	Crypto Express3 Coprocessor	PKCSOAEP with the SHA-256 hash method requires the Sep. 2011 or later licensed internal code (LIC).

Symmetric Key Generate (CSNDSYG and CSNFSYG)

Use the symmetric key generate callable service to generate an AES or DES DATA key and return the key in two forms: enciphered under the master key and encrypted under an RSA public key.

You can import the RSA public key encrypted form by using the symmetric key import service at the receiving node.

Also use the symmetric key generate callable service to generate any DES importer or exporter key-encrypting key encrypted under a RSA public key according to the PKA92 formatting structure.

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

Format

```
CALL CSNDSYG(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_encrypting_key_identifier,
             RSA_public_key_identifier_length,
             RSA_public_key_identifier,
             local_enciphered_key_token_length,
             local_enciphered_key_token,
             RSA_enciphered_key_length,
             RSA_enciphered_key)
```

Parameters

return_code

Direction: Output Type: Integer

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

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.

exit_data_length

Direction: Input/Output Type: Integer

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

exit_data

Direction: Input/Output Type: String

The data that is passed to the installation exit.

rule_array_count

Direction: Input Type: Integer

The number of keywords you supplied in the *rule_array* parameter. The value must be 1, 2, 3, 4, 5, 6, or 7.

rule_array

Direction: Input Type: String

Keywords that provide control information to the callable service. Table 4 lists the keywords. The keywords must be 8 bytes of contiguous storage with the keyword left-justified in its 8-byte location and padded on the right with blanks.

Table 4. Keywords for Symmetric Key Generate Control Information

Keyword	Description	Algorithm
Algorithm (one keyword, optional)		
AES	The key being generated is a secure AES key.	AES
DES	The key being generated is a DES key. This is the default.	DES
Key formatting method (one keyword required)		

Table 4. Keywords for Symmetric Key Generate Control Information (continued)

Keyword	Description	Algorithm
PKA92	Specifies the key-encrypting key is to be encrypted under a PKA96 RSA public key according to the PKA92 formatting structure.	DES
PKCSOAEP	Specifies using the method found in RSA DSI PKCS #1V2 OAEP. The default hash method is SHA-1. Use the SHA-256 keyword for the SHA-256 hash method.	AES or DES
PKCS-1.2	Specifies the method found in RSA DSI PKCS #1 block type 02.	AES or DES
ZERO-PAD	The clear key is right-justified in the field provided, and the field is padded to the left with zeros up to the size of the RSA encryption block (which is the modulus length).	AES or DES
Key Length (optional - for use with PKA92)		
SINGLE-R	For key-encrypting keys, this specifies that the left half and right half of the generated key will have identical values. This makes the key operate identically to a single-length key with the same value. Without this keyword, the left and right halves of the key-encrypting key will each be generated randomly and independently.	DES
Key Length (optional - for use with PKCSOAEP, PKCS-1.2, or ZERO-PAD)		
SINGLE, KEYLN8	Specifies that the generated key should be 8 bytes in length.	DES
DOUBLE	Specifies that the generated key should be 16 bytes in length.	DES
KEYLN16	Specifies that the generated key should be 16 bytes in length.	AES or DES
KEYLN24	Specifies that the generated key should be 24 bytes in length.	AES or DES
KEYLN32	Specifies that the generated key should be 32 bytes in length.	AES
Encipherment method for the local enciphered copy of the key (optional - for use with PKCSOAEP, PKCS-1.2, or ZERO-PAD		
OP	Enciphers the key with the master key. The DES master key is used with DES keys and the AES master key is used with AES keys.	AES or DES
EX	Enciphers the key with the EXPORTER key that is provided through the *key_encrypting_key_identifier* parameter.	DES

Table 4. Keywords for Symmetric Key Generate Control Information (continued)

Keyword	Description	Algorithm
IM	Enciphers the key with the IMPORTER key-encrypting key specified with the *key_encrypting_key_identifier* parameter.	DES
Key Wrapping Method (optional)		
USECONFG	Specifies that the system default configuration should be used to determine the wrapping method. This is the default keyword. The system default key wrapping method can be specified using the DEFAULTWRAP parameter in the installation options data set. See the *z/OS Cryptographic Services ICSF System Programmer's Guide*.	AES and DES
WRAP-ENH	Use enhanced key wrapping method, which is compliant with the ANSI X9.24 standard.	DES
WRAP-ECB	Use original key wrapping method, which uses ECB wrapping for DES key tokens and CBC wrapping for AES key tokens.	AES or DES
Translation Control (optional)		
ENH-ONLY	Restrict rewrapping of the *target_key_identifier* token. Once the token has been wrapped with the enhanced method, it cannot be rewrapped using the original method.	DES
Hash Method (optional - only valid with PKCSOAEP)		
SHA-1	Specifies to use the SHA-1 hash method to calculate the OAEP message hash. This is the default.	AES or DES
SHA-256	Specifies to use the SHA-256 hash method to calculate the OAEP message hash.	AES or DES

key_encrypting_key_identifier

Direction: Input/Output Type: String

The label or internal token of a key-encrypting key. If the *rule_array* specifies IM, this DES key must be an IMPORTER. If the *rule_array* specifies EX, this DES key must be an EXPORTER. Otherwise, the parameter is ignored.

RSA_public_key_identifier_length

Direction: Input Type: Integer

The length of the *RSA_public_key_identifier* parameter. If the *RSA_public_key_identifier* parameter is a label, this parameter specifies the length of the label. The maximum size is 3500 bytes.

RSA_public_key_identifier

Direction: Input Type: String

The token, or label, of the RSA public key to be used for protecting the generated symmetric key.

local_enciphered_key_token_length (was DES_enciphered_key_token_length)

Direction: Input/Output Type: Integer

The length in bytes of the local_enciphered_key_token. This field is updated with the actual length of the token that is generated. The minimum length is 64-bytes and the maximum length is 128 bytes.

local_enciphered_key_token (was DES_enciphered_key_token)

Direction: Input/Output Type: String

This parameter contains the generated DATA key in the form of an internal or external token, depending on rule_array specification. If you specify PKA92, on input specify an internal (operational) key token of an Importer or Exporter Key.

RSA_enciphered_key_length

Direction: Input/Output Type: Integer

The length of the *RSA_enciphered_key* parameter. This service updates this field with the actual length of the *RSA_enciphered_key* it generates. The maximum size is 512 bytes.

RSA_enciphered_key

Direction: Input/Output Type: String

This field contains the RSA enciphered key, which the public key specified in the *RSA_public_key_identifier* field protects.

Restrictions

If the service is executed on the Cryptographic Coprocessor Feature, and you specify IM in the *rule_array*, you must enable Special Secure Mode.

Use of PKA92 or PKCSOAEP requires a PCICC, PCIXCC, CEX2C, or CEX3C.

Usage Notes

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

If the service is executed on the Cryptographic Coprocessor Feature, the generated internal DATA key token is marked according to the system default algorithm.

The hardware configuration sets the limit on the modulus size of keys for key management; thus, this service will fail if the RSA key modulus bit length exceeds this limit.

Specification of PKA92 with an input NOCV key-encrypting key token is not supported.

Use the PKA92 key-formatting method to generate a key-encrypting key. The service enciphers one key copy using the key encipherment technique employed in the IBM Transaction Security System (TSS) 4753, 4755, and AS/400 cryptographic product PKA92 implementations. The control vector for the RSA-enciphered copy of the key is taken from an internal (operational) DES key token that must be present on input in the *RSA_enciphered_key* variable. Only key-encrypting keys that conform to the rules for an OPEX case under the key generate service are permitted. The control vector for the local key is taken from a DES key token that must be present on input in the *local_enciphered_key_token* variable. The control vector for one key copy must be from the EXPORTER class while the control vector for the other key copy must be from the IMPORTER class.

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

Table 5. Symmetric key generate required hardware

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900	Cryptographic Coprocessor Feature	ICSF routes this service to a PCI Cryptographic Coprocessor if one is available on your server. This service will not be routed to a PCI Cryptographic Coprocessor if the modulus bit length of the RSA public key is less than 512 bits. RSA keys with moduli greater than 1024-bit length are not supported. Secure AES keys are not supported. DES, ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 keywords not supported.
	PCI Cryptographic Coprocessor	Use of keyword PKA92 or PKCSOAEP requires the PCI Cryptographic Coprocessor. PKCSOAEP uses the SHA-1 hash method. RSA keys with moduli greater than 2048-bit length are not supported. Secure AES keys are not supported. DES, ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, SHA-1, and SHA-256 keywords not supported. PKCSOAEP with the SHA-256 hash method is not supported.

Table 5. Symmetric key generate required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 990 IBM @server zSeries 890	PCI X Cryptographic Coprocessor Crypto Express2 Coprocessor	The generated internal DATA key will not have any system encryption algorithm markings. RSA keys with moduli greater than 2048-bit length are not supported. Secure AES keys are not supported. ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 keywords not supported. PKCSOAEP with the SHA-256 hash method is not supported.
IBM Systems z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	The generated internal DATA key will not have any system encryption algorithm markings. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Secure AES keys require the Nov. 2008 or later licensed internal code (LIC). ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 not supported. PKCSOAEP with the SHA-256 hash method is not supported.

Table 5. Symmetric key generate required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	The generated internal DATA key will not have any system encryption algorithm markings. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Secure AES keys require the Nov. 2008 or later licensed internal code (LIC). ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 not supported. PKCSOAEP with the SHA-256 hash method is not supported.
	Crypto Express3 Coprocessor	The generated internal DATA key will not have any system encryption algorithm markings. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Secure AES keys require the Nov. 2008 or later licensed internal code (LIC). The SHA-256 keyword is not supported. PKCSOAEP with the SHA-256 hash method is not supported.
z196	Crypto Express3 Coprocessor	The generated internal DATA key will not have any system encryption algorithm markings. PKCSOAEP with the SHA-256 hash method requires the Sep. 2011 or later licensed internal code (LIC).

Symmetric Key Import (CSNDSYI and CSNFSYI)

Use the symmetric key import callable service to import a symmetric AES DATA or DES DATA key enciphered under an RSA public key. It returns the key in operational form, enciphered under the master key.

This service also supports import of a PKA92-formatted DES key-encrypting key under a PKA96 RSA public key.

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

Format

```
CALL CSNDSYI(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             RSA_enciphered_key_length,
             RSA_enciphered_key,
             RSA_private_key_identifier_length,
             RSA_private_key_identifier,
             target_key_identifier_length,
             target_key_identifier)
```

Parameters

return_code

Direction: Output Type: Integer

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

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.

exit_data_length

Direction: Input/Output Type: Integer

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

exit_data

Direction: Input/Output Type: String

The data that is passed to the installation exit.

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, 4, or 5.

rule_array

Direction: Input Type: String

The keywords that provide control information to the callable service. Table 6 on page 17 provides a list. The recovery method is the method to use to

recover the symmetric key. The keywords must be 8 bytes of contiguous storage with the keyword left-justified in its 8-byte location and padded on the right with blanks.

Table 6. Keywords for Symmetric Key Import Control Information

Keyword	Meaning
Algorithm (one keyword, optional)	
AES	The key being imported is an AES key.
DES	The key being imported is a DES key. This is the default.
Recovery Method (required)	
PKA92	Supported by the DES algorithm. Specifies the key-encrypting key is encrypted under a PKA96 RSA public key according to the PKA92 formatting structure.
PKCSOAEP	Specifies to use the method found in RSA DSI PKCS #1V2 OAEP. Supported by the DES and AES algorithms. The default hash method is SHA-1. Use the SHA-256 keyword for the SHA-256 hash method.
PKCS-1.2	Specifies to use the method found in RSA DSI PKCS #1 block type 02. Supported by the DES and AES algorithms.
ZERO-PAD	The clear key is right-justified in the field provided, and the field is padded to the left with zeros up to the size of the RSA encryption block (which is the modulus length). Supported by the DES and AES algorithms.
Key Wrapping Method (optional)	
USECONFG	Specifies that the system default configuration should be used to determine the wrapping method. This is the default keyword. The system default key wrapping method can be specified using the DEFAULTWRAP parameter in the installation options data set. See the *z/OS Cryptographic Services ICSF System Programmer's Guide*.
WRAP-ENH	Use enhanced key wrapping method, which is compliant with the ANSI X9.24 standard.
WRAP-ECB	Use original key wrapping method, which uses ECB wrapping for DES key tokens and CBC wrapping for AES key tokens.
Translation Control (optional)	
ENH-ONLY	Restrict rewrapping of the *target_key_identifier* token. Once the token has been wrapped with the enhanced method, it cannot be rewrapped using the original method.
Hash Method (optional - only valid with PKCSOAEP)	
SHA-1	Specifies to use the SHA-1 hash method to calculate the OAEP message hash. This is the default.
SHA-256	Specifies to use the SHA-256 hash method to calculate the OAEP message hash.

RSA_enciphered_key_length

Direction: Input Type: integer

The length of the *RSA_enciphered_key* parameter. The maximum size is 512 bytes.

RSA_enciphered_key

Direction: Input Type: String

The key to import, protected under an RSA public key. The encrypted key is in the low-order bits (right-justified) of a string whose length is the minimum number of bytes that can contain the encrypted key. This string is left-justified within the *RSA_enciphered_key* parameter.

RSA_private_key_identifier_length

Direction: Input Type: Integer

The length of the *RSA_private_key_identifier* parameter. When the *RSA_private_key_identifier* parameter is a key label, this field specifies the length of the label. The maximum size is 3500 bytes.

RSA_private_key_identifier

Direction: Input Type: String

An internal RSA private key token or label whose corresponding public key protects the symmetric key.

target_key_identifier_length

Direction: Input/Output Type: Integer

The length of the *target_key_identifier* parameter. This field is updated with the actual length of the *target_key_identifier* that is generated. The size must be 64 bytes.

target_key_identifier

Direction: Input/Output Type: String

This field contains the internal token of the imported symmetric key. Except for PKA92 processing, this service produces a DATA key token with a key of the same length as that contained in the imported token.

Restrictions

The exponent of the RSA public key must be odd.

Usage Notes

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

If the service is executed on the Cryptographic Coprocessor Feature, the generated internal DATA key token is marked according to the default system encryption

algorithm unless token copying overrides this. Token copying is accomplished by supplying a valid DATA token with the desired algorithm marks in the *target_key_identifier* field.

The hardware configuration sets the limit on the modulus size of keys for key management; thus, this service will fail if the RSA key modulus bit length exceeds this limit. The service will fail with return code 12 and reason code 11020.

Specification of PKA92 with an input NOCV key-encrypting key token is not supported.

During initialization of a PCICC, PCIXCC, CEX2C, or CEX3C, an Environment Identification, or EID, of zero will be set in the coprocessor. This will be interpreted by the PKA Symmetric Key Import service to mean that environment identification checking is to be bypassed. Thus it is possible on a OS/390 system for a key-encrypting key RSA-enciphered at a node (EID) to be imported at the same node.

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

Symmetric Key Import

Table 7. Symmetric key import required hardware

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900	Cryptographic Coprocessor Feature	Request routed to the CCF when - • The *RSA_private_key_identifier* is a modulus-exponent form private key with a private section ID of X'02' • The key modulus bit length is less than 512 RSA keys with moduli greater than 1024-bit length are not supported. Encrypted AES keys are not supported. DES, ENH-ONLY, USECONFG, WRAP-ENH and WRAP-ECB keywords not supported.
	PCI Cryptographic Coprocessor	Request routed to PCICC when • The *RSA_private_key_identifier* is a modulus-exponent form private key with a private section ID of X'06' • The *RSA_private_key_identifier* is a CRT form private key with a private section ID of X'08' • The *RSA_private_key_identifier* is a retained key • PKA92 recovery method specified • PKCSOAEP recovery method (which uses the SHA-1 hash method) specified RSA keys with moduli greater than 2048-bit length are not supported. Encrypted AES keys are not supported. DES, ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 keywords not supported. PKCSOAEP with the SHA-256 hash method is not supported.

Table 7. Symmetric key import required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 990 IBM @server zSeries 890	PCI X Cryptographic Coprocessor Crypto Express2 Coprocessor	The imported internal DATA key will not have any system encryption markings. Old RSA private keys encrypted under the CCF KMMK is not usable if the KMMK is not the same as the PCIXCC/CEX2C ASYM-MK. RSA keys with moduli greater than 2048-bit length are not supported. Encrypted AES keys are not supported. ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 keywords not supported. PKCSOAEP with the SHA-256 hash method is not supported.
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	The imported internal DATA key will not have any system encryption markings. Old RSA private keys encrypted under the CCF KMMK is not usable if the KMMK is not the same as the CEX2C ASYM-MK. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Encrypted AES keys are not supported. ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 keywords not supported. PKCSOAEP with the SHA-256 hash method is not supported.

Table 7. Symmetric key import required hardware (continued)

Server	Required cryptographic hardware	Restrictions
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	The imported internal DATA key will not have any system encryption markings. Old RSA private keys encrypted under the CCF KMMK is not usable if the KMMK is not the same as the CEX2C or CEX3C ASYM-MK. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Encrypted AES key support requires the Nov. 2008 or later licensed internal code (LIC). ENH-ONLY, USECONFG, WRAP-ENH, WRAP-ECB, and SHA-256 keywords not supported. PKCSOAEP with the SHA-256 hash method is not supported.
	Crypto Express3 Coprocessor	The imported internal DATA key will not have any system encryption markings. Old RSA private keys encrypted under the CCF KMMK is not usable if the KMMK is not the same as the CEX2C or CEX3C ASYM-MK. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC). Encrypted AES key support requires the Nov. 2008 or later licensed internal code (LIC). The SHA-256 keyword is not supported. PKCSOAEP with the SHA-256 hash method is not supported.
z196	Crypto Express3 Coprocessor	The imported internal DATA key will not have any system encryption markings. Old RSA private keys encrypted under the CCF KMMK is not usable if the KMMK is not the same as the CEX2C or CEX3C ASYM-MK. PKCSOAEP with the SHA-256 hash method requires the Sep. 2011 or later licensed internal code (LIC).

Reason Codes for Return Code 8 (8)

A return code of 8 indicates that the call to the service was unsuccessful. The following reason code has been added:

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

Reason Code Hex (Decimal)	Description
081 (129)	A Required Rule Array keyword was not specified. **User action:** Refer to the *rule_array* parameter described in this publication under the appropriate callable service for the correct value.