z/OS
Cryptographic Services
Integrated Cryptographic Service Facility

PKA Key Translate – APAR OA28000

(Updated October 19, 2009)

Contents

Chapter 1. Overview

This book update contains alterations to information previously presented in the *ICSF Application Programmer's Guide*, SA22-7522-12, and the *ICSF Administrator's Guide*, SA22–7521–13, which support z/OS Version 1 Release 10.

Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

These updates relate to the enhancements made to the ICSF product by the application of APAR OA28000. This APAR requires:

- IBM System z9 with driver D67L, plus MCL bundle 42B (available in April 2009), or
- IBM System z10 with driver D76D, plus MCL bundle 23A (available in April 2009).

Chapter 2. PKA Key Generate (CSNDPKG and CSNFPKG)

Use the PKA key generate callable service to generate these PKA keys:

- PKA internal tokens for use with the DSS algorithm in the digital signature services
- RSA keys for use on the Cryptographic Coprocessor Feature, PCI Cryptographic Coprocessor, or PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor.

Input to the PKA key generate callable service is either a skeleton key token that has been built by the PKA key token build service or a valid internal token. In the case of a valid internal token, PKG will generate a key with the same modulus length and the same exponent. Internal tokens with a X'09' section are not supported.

DSS key generation requires this information in the input skeleton token:

- Size of modulus p in bits
- Prime modulus p
- Prime divisor q
- Public generator g
- Optionally, the private key name

DSS standards define restrictions on p, q, and g. (Refer to the Federal Information Processing Standard (FIPS) Publication 186 for DSS standards.) This callable service does not verify all of these restrictions. If you do not follow these restrictions, the keys you generate may not be valid DSS keys. The PKA Key Token Build service or an existing internal or external PKA DSS token can generate the input skeleton token, but all of the preceding must be provided. You can extract the DSS public key token from the internal private key token by calling the PKA public key extract callable service.

Note: DSS keys are not supported on a PCIXCC/CEX2C.

RSA key generation requires this information in the input skeleton token:

- Size of the modulus in bits. The modulus for modulus-exponent form keys is between 512 and 1024. The CRT modulus is between 512 and 4096. The modulus for the variable-length-modulus-exponent form is between 512 and 4096.

RSA key generation has these restrictions: For modulus-exponent, there are restrictions on modulus, public exponent, and private exponent. For CRT, there are restrictions on dp, dq, U, and public exponent. See the Key value structure in Chapter 3, “PKA Key Token Build (CSNDPKB and CSNFPKB),” on page 9 for a summary of restrictions.

Note: The Transaction Security System PKA96 PKA key generate verb supports RSA key generation only; it does not support DSS key generation.

This callable service supports invocation in AMODE(64). The callable service name for AMODE(64) invocation is CSNFPKG.

Format

```
CALL CSNDPKG(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             regeneration_data_length,
             regeneration_data,
             skeleton_key_identifier_length,
             skeleton_key_identifier,
             transport_key_identifier,
             generated_key_token_length,
             generated_key_token)
```

Parameters

return_code

Direction: Output Type: Integer

The return code specifies the general result of the callable service. Chapter 5, “ICSF and TSS Return and Reason Codes” 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. Chapter 5, “ICSF and TSS Return and Reason Codes” lists the reason codes.

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. Value may be 1 or 2.

rule_array

Direction: Input Type: String

A keyword that provides control information to the callable service. See Table 1 for a list. A keyword is left-justified in an 8-byte field and padded on the right with blanks.

Table 1. Keywords for PKA Key Generate Rule Array

Keyword	Meaning
Private Key Encryption (required)	
CLEAR	Return the private key in clear text. The private key in clear text is an external token. Only valid for RSA keys.
MASTER	Encipher the private key under the master key. The keyword is not supported if a skeleton token with a X'09' section is provided.
RETAIN	Retain the private key within the PCI Cryptographic Coprocessor for additional security. Only valid for RSA signature keys. The keyword is not supported if a skeleton token with a X'09' section is provided.
XPORT	Encipher the private key under the *transport_key_identifier*. Only valid for RSA keys.
Options (optional)	
CLONE	Mark a generated and retained private key as usable in cryptographic engine cloning process. This keyword is supported only if RETAIN is also specified. Only valid for RSA keys. The keyword is not supported if a skeleton token with a X'09' section is provided.

regeneration_data_length

Direction: Input Type: Integer

The value must be 0 for DSS tokens. For RSA tokens, the regeneration_data_length can be non-zero. If it is non-zero, it must be between 8 and 512 bytes inclusive.

regeneration_data

Direction: Input Type: String

This field points to a string variable containing a string used as the basis for creating a particular public-private key pair in a repeatable manner.

skeleton_key_identifier_length

Direction: Input Type: Integer

The length of the *skeleton_key_identifier* parameter in bytes. The maximum allowed value is 3500 bytes.

skeleton_key_identifier

Direction: Input Type: String

The application-supplied skeleton key token generated by PKA key token build or label of the token that contains the required network quantities for DSS key generation, or the required modulus length and public exponent for RSA key

generation. If RETAIN was specified and the *skeleton_key_identifier* is a label, the label must match the private key name of the key.

The skeleton_key_identifier parameter must contain a token which specifies a modulus length in the range 512 – 4096 bits.

transport_key_identifier

Direction: Input Type: String

A 64-byte field to contain a DES key identifier. This field must be binary zeros, unless the XPORT rule is specified. For XPORT rule, this is an IMPORTER or EXPORTER key or the label of an IMPORTER or EXPORTER key that is used to encrypt the generated key. If you specify a label, it must resolve uniquely to either an IMPORTER or EXPORTER key. Only valid for RSA keys.

generated_key_token_length

Direction: Input/Output Type: Integer

The length of the generated key token. The field is checked to ensure it is at least equal to the token being returned. The maximum size is 3500 bytes. On output, this field is updated with the actual token length.

generated_key_token

Direction: Input/Output Type: String

The internal token or label of the generated DSS or RSA key. The label can be that of a retained key. Checks are made to ensure that a retained key is not overlayed in PKDS. If the label is that of a retained key, the private name in the token must match the label name. If a label is specified in the *generated_key_token* field, the *generated_key_token_length* returned to the application will be the same as the input length. If RETAIN was specified, but the *generated_key_token* was not specified as a label, the generated key length returned to the application will be zero (the key was retained in the PCI Cryptographic Coprocessor). If the record already exists in the PKDS with the same label as the one specified as the *generated_key_token*, the record will be overwritten with the newly generated key token (unless the PKDS record is an existing retained private key, in which case it cannot be overwritten). If there is no existing PKDS record with this label in the case of generating a retained key, a record will be created. For generation of a non-retained key, if a label is specified in the *generated_key_token* field, a record must already exist in the PKDS with this same label or the service will fail.

Restriction

Caller must be task mode and must not be SRB mode, when running on z900/z800 servers.

The 2048-bit RSA keys may have an exponent in the range of 0-256. The 4096-bit RSA key exponents are restricted to the values 3 and 65537.

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.

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

Table 2. PKA key generate required hardware

Server	Required Cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900	Cryptographic Coprocessor Feature	The service examines the skeleton token and routes the generation request to the appropriate cryptographic processor. If the skeleton is a DSS key token, processing takes place on the Cryptographic Coprocessor Feature.
	PCI Cryptographic Coprocessor	The service examines the skeleton token and routes the generation request to the appropriate cryptographic processor. If the skeleton is a DSS key token, processing takes place on the Cryptographic Coprocessor Feature.
IBM @server zSeries 990 IBM @server zSeries 890	PCI X Cryptographic Coprocessor Crypto Express2 Coprocessor	DSS tokens are not supported. RSA keys with moduli greater than 2048-bit length are not supported.
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	DSS tokens are not supported. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC).
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	DSS tokens are not supported. RSA key support with moduli within the range 2048-bit to 4096-bit requires the Nov. 2007 or later licensed internal code (LIC).

Chapter 3. PKA Key Token Build (CSNDPKB and CSNFPKB)

Use this callable service to build external PKA key tokens containing unenciphered private RSA or DSS keys, or public RSA or DSA keys. This callable service is used to create the following:

- A skeleton_key_token for use with the PKA Key Generate callable service (see Table 1 on page 5)
- A key token with a public key that has been obtained from another source
- A key token with a clear private-key and the associated public key
- A key token for an RSA private key in optimized Chinese Remainder Theorem (CRT) form.
- An RSA token with X'09' section identifier using the RSAMEVAR keyword to obtain a token for a key in modulus-exponent form that is variable length.

DSS key generation requires this information in the input skeleton token:

- Size of modulus p in bits
- Prime modulus p
- Prime divisor q
- Public generator g
- Optionally, the private key name

Note: DSS standards define restrictions on the prime modulus p, prime divisor q, and public generator g. (Refer to the Federal Information Processing Standard (FIPS) Publication 186 for DSS standards.) This callable service does not verify all of these restrictions. If you do not follow the restrictions, the keys you generate may not be valid DSS keys.

Restriction: DSS is not supported on a PCIXCC/CEX2C. PKA key token build will still build DSS tokens, but they cannot be used in any other service on the z890, z990, z9 EC, z9 BC, z10 EC and z10 BC.

This callable service supports invocation in AMODE(64). The callable service name for AMODE(64) invocation is CSNFPKB.

Format

```
CALL CSNDPKB(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             key_value_structure_length,
             key_value_structure,
             private_key_name_length,
             private_key_name,
             reserved_1_length,
             reserved_1,
             reserved_2_length,
             reserved_2,
             reserved_3_length,
             reserved_3,
             reserved_4_length,
             reserved_4,
             reserved_5_length,
             reserved_5,
             key_token_length,
             key_token)
```

Parameters

return_code

Direction: Output Type: Integer

The return code specifies the general result of the callable service. Chapter 5, "ICSF and TSS Return and Reason Codes" 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. Chapter 5, "ICSF and TSS Return and Reason Codes" lists the reason codes.

exit_data_length

Direction: Ignored Type: Integer

Reserved field.

exit_data

Direction: Input/Output Type: String

Reserved field.

rule_array_count

Direction: Input Type: Integer

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

rule_array

Direction: Input Type: String

One or two keywords that provide control information to the callable service. Table 3 lists the keywords. The keywords must be in contiguous storage with each of the keywords left-justified in its own 8-byte location and padded on the right with blanks.

Table 3. Keywords for PKA Key Token Build Control Information

Keyword	Meaning
Key Type (required)	
DSS-PRIV	This keyword indicates building a key token containing both public and private DSS key information. The parameter *key_value_structure* identifies the input key values, if supplied.
DSS-PUBL	This keyword indicates building a key token containing public DSS key information. The parameter *key_value_structure* identifies the input key values, if supplied.
RSA-CRT	This keyword indicates building a token containing an RSA private key in the optimized Chinese Remainder Theorem (CRT) form. The parameter *key_value_structure* identifies the input key values, if supplied.
RSA-PRIV	This keyword indicates building a token containing both public and private RSA key information. The parameter *key_value_structure* identifies the input key values, if supplied.
RSA-PUBL	This keyword indicates building a token containing public RSA key information. The parameter *key_value_structure* identifies the input values, if supplied.
RSAMEVAR	This keyword is for creating a key token for an RSA public and private key pair in modulus-exponent form whose modulus is 512 bits or greater.
Key Usage Control (optional)	
KEY-MGMT	Indicates that an RSA private key can be used in both the symmetric key import and the digital signature generate callable services.
KM-ONLY	Indicates that an RSA private key can be used only in symmetric key distribution.
SIG-ONLY	Indicates that an RSA private key cannot be used in symmetric key distribution. This is the default. Note that for DSS-PRIV the keyword is allowed but extraneous; DSS keys are defined only for digital signature.
Translate Control (optional)	

Table 3. Keywords for PKA Key Token Build Control Information (continued)

Keyword	Meaning
XLATE-OK	Specifies that the private key material can be translated. XLATE-OK is only allowed with key types RSA-PRIV, RSAMEVAR, RSA-CRT and is valid with all key usage rules.
NO-XLATE	Indicates key translation is not allowed. This is the default. NO-XLATE is only allowed with key types RSA-PRIV, RSAMEVAR, RSA-CRT and is valid with all key usage rules.

key_value_structure_length

Direction: Input Type: Integer

This is a segment of contiguous storage containing a variable number of input clear key values. The length depends on the key type parameter in the rule array and on the actual values input. The length is in bytes.

Table 4. Key Value Structure Length Maximum Values for Key Types

Key Type	Key Value Structure Maximum Value
DSS-PRIV	436
DSS-PUBL	416
RSA-CRT	3500
RSAMEVAR	3500
RSA-PRIV	648
RSA-PUBL	520

key_value_structure

Direction: Input Type: String

This is a segment of contiguous storage containing a variable number of input clear key values and the lengths of these values in bits or bytes, as specified. The structure elements are ordered, of variable length, and the input key values must be right-justified within their respective structure elements and padded on the left with binary zeros. If the leading bits of the modulus are zero's, don't count them in the length. Table 5 defines the structure and contents as a function of key type.

Table 5. Key Value Structure Elements for PKA Key Token Build

Offset	Length (bytes)	Description
Key Value Structure (Optimized RSA, Chinese Remainder Theorem form, RSA-CRT)		
000	002	Modulus length in bits (512 to 4096). This is required.
002	002	Modulus field length in bytes, "nnn." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. This value must not exceed 512.

Table 5. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
004	002	Public exponent field length in bytes, "eee." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service.
006	002	Reserved, binary zero.
008	002	Length of the prime number, p, in bytes, "ppp." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of p + q is 512 bytes.
010	002	Length of the prime number, q, in bytes, "qqq." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of p + q is 512 bytes.
012	002	Length of d_p, in bytes, "rrr." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of d_p + d_q is 512 bytes.
014	002	Length of d_q, in bytes, "sss." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of d_p + d_q is 512 bytes.
016	002	Length of U, in bytes, "uuu." This value can be zero if the key token is used as a *skeleton_key_token* in the PKA key generate callable service. Maximum size of U is 512 bytes.
018	nnn	Modulus, n.
018 + nnn	eee	Public exponent, e. This is an integer such that 1<e<n. e must be odd. When you are building a *skeleton_key_token* to control the generation of an RSA key pair, the public key exponent can be one of these values: 3, 65537 (2^{16} + 1), or 0 to indicate that a full random exponent should be generated. The exponent field can be a null-length field if the exponent value is 0.
018 + nnn + eee	ppp	Prime number, p.
018 + nnn + eee + ppp	qqq	Prime number, q.

Table 5. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
018 + nnn + eee + ppp + qqq	rrr	d_p = d mod(p-1).
018 + nnn + eee + ppp + qqq + rrr	sss	d_q = d mod(q-1).
018 + nnn + eee + ppp + qqq + rrr + sss	uuu	U = q^{-1}mod(p).
Key Value Structure (RSA Private, RSA Private variable or RSA Public)		
000	002	Modulus length in bits. This is required. When building a skeleton token, the modulus length in bits must be greater than or equal to 512 bits.
002	002	Modulus field length in bytes, "XXX". This value can be zero if you are using the key token as a skeleton in the PKA key generate verb. This value must not exceed 512 when either the RSA-PUBL or RSAMEVAR keyword is used, and must not exceed 128 when the RSA-PRIV keyword is used. This service can build a key token for a public RSA key with a 4096-bit modulus length, or it can build a key token for a 1024-bit modulus length private key.
004	002	Public exponent field length in bytes, "YYY". This value must not exceed 512 when either the RSA-PUBL or RSAMEVAR keyword is used, and must not exceed 128 when the RSA-PRIV keyword is used. This value can be zero if you are using the key token as a skeleton token in the PKA key generate verb. In this case, a random exponent is generated. To obtain a fixed, predetermined public key exponent, you can supply this field and the public exponent as input to the PKA key generate verb.
006	002	Private exponent field length in bytes, "ZZZ". This field can be zero, indicating that private key information is not provided. This value must not exceed 128 bytes. This value can be zero if you are using the key token as a skeleton token in the PKA key generate verb.

Table 5. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
008	XXX	Modulus, n. This is an integer such that 1 < n <2**2048.The n is the product of p and q for primes p and q.
008 + XXX	YYY	RSA public exponent, e. This is an integer such that 1<e<n. e must be odd. When you are building a *skeleton_key_token* to control the generation of an RSA key pair, the public key exponent can be one of these values: 3, 65537 (2^{16} + 1), or 0 to indicate that a full random exponent should be generated. The exponent field can be a null-length field if the exponent value is 0.
008 + XXX + YYY	ZZZ	RSA secret exponent d. This is an integer such that 1<d<n. The value of d is e^{-1} mod(p-1)(q-1); the This can be a null-length field if you are using the key token as a skeleton token in the PKA key generate verb.
Key Value Structure (DSS Private or DSS Public)		
000	002	Modulus length in bits. This is required.
002	002	Prime modulus field length in bytes, "XXX". You can supply this as a network quantity to the ICSF PKA key generate callable service, which uses the quantity to generate DSS keys. The maximum allowed value is 128.
004	002	Prime divisor field length in bytes, "YYY". You can supply this as a network quantity to the ICSF PKA key generate callable service, which uses the quantity to generate DSS keys. The allowed values are 0 or 20 bytes.
006	002	Public generator field length in bytes, "ZZZ". You can supply this in a skeleton token as a network quantity to the ICSF PKA key generate callable service, which uses the quantity to generate DSS keys. The maximum allowed value is 128 bytes and is exactly the same length as the prime modulus.

Table 5. Key Value Structure Elements for PKA Key Token Build (continued)

Offset	Length (bytes)	Description
008	002	Public key field length in bytes, "AAA". This field can be zero, indicating that the ICSF PKA key generate callable service generates a value at random from supplied or generated network quantities. The maximum allowed value is 128 bytes and is exactly the same length as the prime modulus.
010	002	Secret key field length in bytes, "BBB". This field can be zero, indicating that the ICSF PKA key generate callable service generates a value at random from supplied or generated network quantities. The allowed values are 0 or 20 bytes.
012	XXX	DSS prime modulus p. This is an integer such that $2^{L-1}<p<2^{L}$. The p must be prime. You can supply this value in a skeleton token as a network quantity; it is used in the algorithm that generates DSS keys.
012 + XXX	YYY	DSS prime divisor q. This is an integer that is a prime divisor of p-1 and $2^{159}<q<2^{160}$. You can supply this value in a skeleton token as a network quantity; it is used in the algorithm that generates DSS keys.
012 + XXX+ YYY	ZZZ	DSS public generator g. This is an integer such that 1<g<p. You can supply this value in a skeleton token as a network quantity; it is used in the algorithm that generates DSS keys.
012 + XXX+ YYY+ ZZZ	AAA	DSS public key y. This is an integer such that $y = g^{x} \bmod p$.
012 + XXX+ YYY+ ZZZ+ AAA	BBB	DSS secret private key x. This is an integer such that 0<x<q. The x is random. You need not supply this value if you specify DSS-PUBL in the rule array.

Notes:

1. All length fields are in binary.
2. All binary fields (exponent, lengths, modulus, and so on) are stored with the high-order byte field first. This integer number is right-justified within the key structure element field.
3. You must supply all values in the structure to create a token containing an RSA or DSS private key for input to the PKA key import service.

private_key_name_length

Direction: Input Type: Integer

The length can be 0 or 64.

private_key_name

Direction: Input Type: EBCDIC character

This field contains the name of a private key. The name must conform to ICSF label syntax rules. That is, allowed characters are alphanumeric, national (@,#,$) or period (.). The first character must be alphabetic or national. The name is folded to upper case and converted to ASCII characters. ASCII is the permanent form of the name because the name should be independent of the platform. The name is then cryptographically coupled with clear private key data prior to its encryption of the private key. Because of this coupling, the name can never change when the key token is already imported. The parameter is not valid with key types DSS-PUBL or RSA-PUBL.

reserved_1_length

Direction: Input Type: Integer.

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_1

Direction: Input Type: String

The *reserved_1* parameter identifies a string that is reserved. The service ignores it.

reserved_2_length

Direction: Input Type: Integer.

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_2

Direction: Input Type: String

The *reserved_2* parameter identifies a string that is reserved. The service ignores it.

reserved_3_length

Direction: Input Type: Integer.

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_3

Direction: Input Type: String

The *reserved_3* parameter identifies a string that is reserved. The service ignores it.

reserved_4_length

Direction: Input Type: Integer.

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_4

Direction: Input Type: String

The *reserved_4* parameter identifies a string that is reserved. The service ignores it.

reserved_5_length

Direction: Input Type: Integer.

Length in bytes of a reserved parameter. You must set this variable to 0.

reserved_5

Direction: Input Type: String

The *reserved_5* parameter identifies a string that is reserved. The service ignores it.

key_token_length

Direction: Input/Output Type: Integer

Length of the returned key token. The service checks the field to ensure it is at least equal to the size of the token to return. On return from this service, this field is updated with the exact length of the *key_token* created. On input, a size of 3500 bytes is sufficient to contain the largest *key_token* created.

key_token

Direction: Output Type: String

The returned key token containing an unenciphered private or public key. The private key is in an external form that can be exchanged with different Common Cryptographic Architecture (CCA) PKA systems. You can use the public key token directly in appropriate ICSF signature verification or key management services.

Restrictions

None.

Usage Notes

If you are building a skeleton for use in a PKA Key Generate request to generate a retained PKA private key, you must build a private key name section in the skeleton token.

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

Table 6. PKA key token build required hardware

Server	Required cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900	None.	
IBM @server zSeries 990 IBM @server zSeries 890	None.	
IBM System z9 EC IBM System z9 BC	None.	
IBM System z10 EC IBM System z10 BC	None.	

Chapter 4. PKA Key Translate (CSNDPKT and CSNFPKT)

Use the PKA key translate callable service to translate a source CCA RSA key token into a target external smart card key token.

The source CCA RSA key token must be wrapped with a transport key encrypting key (KEK). The XLATE bit must also be turned on in the key usage byte of the source token. The source token is unwrapped using the specified source transport KEK. The target key token will be wrapped with the specified target transport KEK. Existing information in the target token is overwritten.

There are restrictions on which type key can be used for the source and target transport key tokens. These restrictions are enforced by access control points.

There are restrictions on which rule can be used. These restrictions are enforced by access control points.

This callable service supports invocation in AMODE(64). The callable service name for AMODE(64) invocation is CSNFPKT.

The CSNDPKT and CSNFPKT entry points are also available for C applications using the three CSF DLLs (CSFDLL31, CSFDLL3X, CSFDLL64).

Format

```
CALL CSNDPKT(
             return_code,
             reason_code,
             exit_data_length,
             exit_data,
             rule_array_count,
             rule_array,
             source_key_identifier_length,
             source_key_identifier,
             source_transport_key_identifier_length,
             source_transport_key_identifier,
             target_transport_key_identifier_length,
             target_transport_key_identifier,
             target_key_token_length
             target_key_token)
```

Parameters

return_code

Direction: Output Type: Integer

The return code specifies the general result of the callable service. Chapter 5, "ICSF and TSS Return and Reason Codes" 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. Chapter 5, "ICSF and TSS Return and Reason Codes" lists the reason codes.

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. Value must be 1.

rule_array

Direction: Input Type: String

The smartcard format rule for the callable service. A keyword that provides control information to the callable service. See Table 7 for a list. A keyword is left-justified in an 8-byte field and padded on the right with blanks.

Table 7. Keywords for PKA Key Generate Rule Array

Keyword	Meaning
Smartcard Format (required)	
SCVISA	This keyword indicates translating the key into the smart card Visa proprietary format.
SCCOMME	This keyword indicates translating the key into the smart card Modulus-Exponent format.
SCCOMCRT	This keyword indicates translating the key into the smart card Chinese Remainder Theorem format.

source_key_identifier_length

Direction: Input Type: Integer

Length in bytes of the *source_key_identifier* variable. The maximum length is 3500 bytes.

source_key_identifier

Direction: Input Type: String

This field contains either a key label identifying an RSA private key or an external public-private key token. The private key must be wrapped with a key encrypting key.

source_transport_key_identifier_length

Direction: Input Type: Integer

Length in bytes of the *source_transport_key_identifier* parameter. This value must be 64.

source_transport_key_identifier

Direction: Input/Output Type: String

This field contains an internal token or label of a DES key-encrypting key. This key is used to unwrap the input RSA key token specified with parameter *source_key_identifier*. See "Usage Notes" on page 24 for details on the type of transport key that can be used

target_transport_key_identifier_length

Direction: Input Type: Integer

Length in bytes of the *target_transport_key_identifier* parameter. This value must be 64.

target_transport_key_identifier

Direction: Input/Output Type: String

This field contains an internal token or label of a DES key-encrypting key. This key is used to wrap the output RSA key returned with parameter target_key_token. See "Usage Notes" on page 24 for details on the type of transport key that can be used.

target_key_token_length

Direction: Input/Output Type: Integer

Length in bytes of the *target_key_token* parameter. On output, the value in this variable is updated to contain the actual length of the *target_key_token* produced by the callable service. The maximum length is 3500 bytes.

target_key_token

Direction: Output Type: String

This field contains the RSA key in the smartcard format specified in the rule array and is protected by the key-encrypting key specified in the *target_transport_key* parameter. This is not a CCA token, and cannot be stored in the PKDS.

Restriction

CCA RSA ME tokens will not be translated to the SCCOMCRT format. CCA RSA CRT tokens will not be translated to the SCCOMME format. SCVISA only supports Modulus-Exponent (ME) keys.

Usage Notes

There are access control points that control use of the format rule array keys and the type of transport keys that can be used. All of these access control points are enabled in the default role.

PKA Key Translate - from CCA RSA to SCVISA Format
PKA Key Translate - from CCA RSA to SC ME Format
PKA Key Translate - from CCA RSA to SC CRT Format
PKA Key Translate - from source EXP KEK to target EXP KEK
PKA Key Translate - from source IMP KEK to target EXP KEK
PKA Key Translate - from source IMP KEK to target IMP KEK

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

Table 8. PKA key translate required hardware

Server	Required Cryptographic hardware	Restrictions
IBM @server zSeries 800 IBM @server zSeries 900		Not supported on this platform.
IBM @server zSeries 990 IBM @server zSeries 890		Not supported on this platform.
IBM System z9 EC IBM System z9 BC	Crypto Express2 Coprocessor	Requires the April 2009 or later licensed internal code (LIC).
IBM System z10 EC IBM System z10 BC	Crypto Express2 Coprocessor	Requires the April 2009 or later licensed internal code (LIC).

Chapter 5. ICSF and TSS Return and Reason Codes

Return Codes and Reason Codes

This topic describes return codes and reason codes.

The TSS return and reason codes have been merged with the ICSF codes in this release. If there is a REASONCODES line in the description, it will indicate an alternate reason code you should investigate.

Each return code returns unique reason codes to your application program. The reason codes associated with each return code are described in these topics. The reason code tables present the hexadecimal code followed by the decimal code in parenthesis.

Reason Codes for Return Code 8 (8)

Table 9 lists reason codes returned from callable services that give return code 8.

Most of these reason codes indicate that the call to the service was unsuccessful. No cryptographic processing took place. Therefore, no output parameters were filled. Exceptions to this are noted in the descriptions.

Table 9. Reason Codes for Return Code 8 (8)

Reason Code Hex (Decimal)	**Description**
040 (064)	The supplied private key can be used only for digital signature. Key management services are disallowed. **User action**: Supply a key with key management enabled. OR This service requires an RSA private key that is for signature use. The specified key may be used for key management purposes only. **User action**: Re-invoke the service with a supported private key. OR This service requires an RSA private key that is translatable. The specified key may not be used in the PKA Key Translate callable service. **User action**: Re-invoke the service with a supported private key. To make a key translatable, XLATE-OK must be turned on.
7E0 (2016)	The *rule_array* parameter contents are incorrect. One or more of the rules specified are not valid for this service OR some of the rules specified together may not be combined. **User action**: Refer to the *rule_array* parameter described in this publication under the appropriate callable service for the correct value.
BE7 (3017)	A clear key was provided when a secure key was required. **User action**: Correct the appropriate key identifier.

Chapter 6. Access Control Points and Callable Services

The TKE workstation allows you to enable or disable callable service access control points. For systems that do not use the optional TKE Workstation, all access control points (current and new) are enabled in the DEFAULT Role with the appropriate licensed internal code on the PCI Cryptographic Coprocessor or PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor.

TKE Version 4.0 and higher

Access to services that are executed on the PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor is through Access Control Points in the DEFAULT Role. To execute callable services on the PCI X Cryptographic Coprocessor/Crypto Express2 Coprocessor, access control points must be enabled for each service in the DEFAULT Role.

New TKE users and non-TKE users have all access control points enabled. This is also true for new TKE V5.x users. If you are migrating from TKE V4.0, V4.1, or V4.2 to TKE V5.0 and have a PCIXCC/CEX2C, all your current access control points will remain the same and any new access control points for ICSF will not be enabled.

Note: Access control points DKYGENKY-DALL and DSG ZERO-PAD unrestricted hash length and PTR enhanced PIN security are always disabled in the DEFAULT role for all customers (TKE and Non-TKE). A TKE Workstation is required to enable these access control points.

Access Control Points for APAR OA28000 are:

- PKA Key Translate - from CCA RSA to SCVISA Format
- PKA Key Translate - from CCA RSA to SCCOMME Format
- PKA Key Translate - from CCA RSA to SCCOMCRT Format
- PKA Key Translate - from source EXP KEK to target EXP KEK
- PKA Key Translate - from source IMP KEK to target EXP KEK
- PKA Key Translate - from source IMP KEK to target IMP KEK

Access Control Points for HCR7751 are:

- Clear New AES Master Key Register (ISPF ACP)
- Load First AES Master Key Part (ISPF ACP)
- Combine AES Master Key Parts (ISPF ACP)
- Set AES Master Key (ISPF ACP)
- Multiple Clear Key Import/Multiple Secure Key Import - AES
- Symmetric Algorithm Encipher - Secure AES
- Symmetric Algorithm Decipher - Secure AES
- Symmetric Key Generate - AES, PKCSOEAP, PKCS- 1.2
- Symmetric Key Generate - AES, ZERO-PAD
- Symmetric Key Import - AES, PKCSOEAP, PKCS-1.2
- Symmetric Key Import - AES, ZERO-PAD
- Symmetric Key Export - AES, PKCSOAEP, PKCS-1.2

- Symmetric Key Export - AES, ZERO-PAD

These access control points require the Nov. 2008 or later licensed internal code (LIC).

Access Control Points for HCR7731 are:

- Remote Key Export - Generate or export a key for use by a CCA node
- Trusted Block Create - Activate an Inactive Trusted Key Block
- Trusted Block Create - Create a Trusted Key Block in Inactive Form
- PKA Key Generate - Permit Regeneration Data
- PKA Key Generate - Permit Regeneration Data for Retained Keys
- PTR Enhanced PIN Security

  Callable services affected by PTR enhanced PIN security:
  - Clear PIN Encrypt - PTR Enhanced PIN Security
  - Clear Pin Generate Alternate - PTR Enhanced PIN Security
  - Encrypted PIN Generate - PTR Enhanced PIN Security
  - Encrypted PIN Translate - PTR Enhanced PIN Security
  - Encrypted PIN Verify - PTR Enhanced PIN Security
  - PIN Change/Unblock - PTR Enhanced PIN Security

Access Control Points for HCR770B are:

- Diversified Key Generate - TDES-XOR
- Diversified Key Generate - TDESEMV2/TDESEMV4
- PIN Change/Unblock - change EMV PIN with OPINENC
- PIN Change/Unblock - change EMV PIN with IPINENC
- Transaction Validation - Generate
- Transaction Validation - Verify CSC-3
- Transaction Validation - Verify CSC-4
- Transaction Validation - Verify CSC-5
- Key Part Import - RETRKPR

Access Control Points for HCR770A are:

- CKDS Conversion Program
- Clear Key Import
- Decipher
- Digital Signature Verify
- DSG ZERO-PAD Unrestricted Hash Length
- Encipher
- Key Part Import - ADD-PART keyword
- Key Part Import - COMPLETE keyword
- NOCV Exporter
- NOCV Importer
- Prohibit Export Extended
- Public Key Encrypt

These access control points are only supported on the PCIXCC/CEX2C.

For the relationship between access control points and callable services, see Table 10.

Callable Service Access Control Points

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

Table 10. Callable service access control points

Access Control Point	Callable Service
*Clear Key Import / Multiple Clear Key Import - DES	CSNBCKI or CSNBCKM
*Clear Key Import / Multiple Clear Key Import - AES	CSNBCKI , CSNBCKM or CSNBSKM
Clear PIN Encrypt	CSNBCPE
Clear PIN Generate - 3624	CSNBPGN
Clear PIN Generate - GBP	CSNBPGN
Clear PIN Generate - VISA PVV	CSNBPGN
Clear PIN Generate - Interbank	CSNBPGN
Clear Pin Generate Alternate - 3624 Offset	CSNBCPA
Clear PIN Generate Alternate - VISA PVV	CSNBCPA
Control Vector Translate	CSNBCVT
Cryptographic Variable Encipher	CSNBCVE
CVV Generate	CSNBCSG
CVV Verify	CSNBCSV
DATAM Key Management Control	CSNBKGN, CSNBKIM, CSNBKEX and CSNBDKG
Data Key Export	CSNBDKX
Data Key Export - Unrestricted	CSNBDKX
Data Key Import	CSNBDKM
Data Key Import - Unrestricted	CSNBDKM
*Decipher - DES	CSNBDEC
Digital Signature Generate	CSNDDSG
*DSG ZERO-PAD restriction lifted	CSNDDSG
*Digital Signature Verify	CSNDDSV
Diversified Key Generate - CLR8–ENC	CSNBDKG
Diversified Key Generate - SESS-XOR	CSNBDKG
Diversified Key Generate - TDES-ENC	CSNBDKG
Diversified Key Generate - TDES-DEC	CSNBDKG
**Diversified Key Generate - TDES-XOR	CSNBDKG
**Diversified Key Generate - TDESEMV2/ TDESEMV4	CSNBDKG
Diversified Key Generate - single length or same halves	CSNBDKG
DKYGENKY - DALL	CSNBDKG
*Encipher - DES	CSNBENC

Table 10. Callable service access control points (continued)

Encrypted PIN Generate - 3624	CSNBEPG
Encrypted PIN Generate - GBP	CSNBEPG
Encrypted PIN Generate - Interbank	CSNBEPG
Encrypted PIN Translate - Translate	CSNBPTR
Encrypted PIN Translate - Reformat	CSNBPTR
Encrypted PIN Verify - 3624	CSNBPVR
Encrypted PIN Verify - GPB	CSNBPVR
Encrypted PIN Verify - VISA PVV	CSNBPVR
Encrypted PIN Verify - Interbank	CSNBPVR
Key Export	CSNBKEX
Key Export - Unrestricted	CSNBKEX
Key Generate - OPIM, OPEX, IMEX, etc.	CSNBKGN
Key Generate - EX, IM, OP	CSNBKGN
Key Generate - CVARs	CSNBKGN
Key Generate - SINGLE-R	CSNBKGN
Key Import	CSNBKIM
Key Import - Unrestricted	CSNBKIM
*Key Part Import - ADD-PART	CSNBKPI
*Key Part Import - COMPLETE	CSNBKPI
Key Part Import - first key part	CSNBKPI
Key Part Import - middle and final	CSNBKPI
Key Part Import - unrestricted	CSNBKPI
Key Part Import - RETRKPR	CSNBKPI
Key Translate	CSNBKTR
MAC Generate	CSNBMGN
MAC Verify	CSNBMVR
*NOCV KEK usage for export-related functions	CSNBKEX, CSNBSKM, and CSNBKGN
*NOCV KEK usage for import-related functions	CSNBKIM, CSNBSKI, CSNBSKM, and CSNBKGN
*PCF CKDS Conversion Program	CSFCONV
**PIN Change/Unblock - change EMV PIN with OPINENC	CSNBPCU
**PIN Change/Unblock - change EMV PIN with IPINENC	CSNBPCU
**PIN Change/Unblock - PTR Enhanced PIN Security	CSNBPCU
PKA Decrypt	CSNDPKD
PKA Encrypt	CSNDPKE
PKA Key Generate	CSNDPKG
PKA Key Generate - Clear	CSNDPKG
PKA Key Generate - Clone	CSNDPKG

Table 10. Callable service access control points (continued)

PKA Key Generate - Permit Regeneration Data	CSNDPKG
PKA Key Generate - Permit Regeneration Data Retain	CSNDPKG
PKA Key Import	CSNDPKI
PKA Key Import - Import an External Trusted Key Block to internal form	CSNDPKI
PKA Key Translate - from CCA RSA to SCVISA Format	CSNDPKT
PKA Key Translate - from CCA RSA to SC ME Format	CSNDPKT
PKA Key Translate - from CCA RSA to SC CRT Format	CSNDPKT
PKA Key Translate - from source EXP KEK to target EXP KEK	CSNDPKT
PKA Key Translate - from source IMP KEK to target EXP KEK	CSNDPKT
PKA Key Translate - from source IMP KEK to target IMP KEK	CSNDPKT
PKA Key Token Change	CSNDKTC
Prohibit Export	CSNBPEX
*Prohibit Export Extended	CSNBPEXX
PTR Enhanced PIN Security	CSNBCPE, CSNBCPA, CSNBEPG, CSNBPTR, CSNBPVR, and CSNBPCU
Remote Key Export - Generate or export a key for use by a non-CCA node	CSNDRKX
Retained Key Delete	CSNDRKD
Retained Key List	CSNDRKL
Secure Key Import - IM	CSNBSKI or CSNBSKM
Secure Key Import - OP	CSNBSKI or CSNBSKM
Secure Messaging for Keys	CSNBSKY
Secure Messaging for PINs	CSNBSPN
SET Block Compose	CSNDSBC
SET Block Decompose	CSNDSBD
SET Block Decompose - PIN ext IPINENC	CSNDSBD
SET Block Decompose - PIN ext OPINENC	CSNDSBD
Symmetric Algorithm Decipher - Secure AES	CSNBSAD or CSNBSAD1
Symmetric Algorithm Encipher - Secure AES	CSNBSAE or CSNBSAE1
Symmetric Key Export - AES, PKCS-1.2	CSNDSYX
Symmetric Key Export - DES, PKCS-1.2	CSNDSYX
Symmetric Key Export - AES, ZERO-PAD	CSNDSYX
Symmetric Key Export - DES, ZERO-PAD	CSNDSYX
Symmetric Key Generate - DES, PKA92	CSNDSYG
Symmetric Key Generate - AES, PKCS-1.2	CSNDSYG
Symmetric Key Generate - DES, PKCS-1.2	CSNDSYG

Table 10. Callable service access control points (continued)

Symmetric Key Generate - AES, ZERO-PAD	CSNDSYG
Symmetric Key Generate - DES, ZERO-PAD	CSNDSYG
Symmetric Key Import - DES, PKA92 KEK	CSNDSYI
Symmetric Key Import - AES, PKCS-1.2	CSNDSYI
Symmetric Key Import - DES, PKCS-1.2	CSNDSYI
Symmetric Key Import - AES, ZERO-PAD	CSNDSYI
Symmetric Key Import - DES, ZERO-PAD	CSNDSYI
**Transaction Validation - Generate	CSNBTRV
**Transaction Validation - Verify CSC-3	CSNBTRV
**Transaction Validation - Verify CSC-4	CSNBTRV
**Transaction Validation - Verify CSC-5	CSNBTRV
Trusted Block Create - Activate an Inactive Trusted Key Block	CSNDTBC
Trusted Block Create - Create Trusted Key Block in Inactive Form	CSNDTBC
UKPT - PIN Verify, PIN Translate	CSNBPVR and CSNBPTR

Notes:

1. * indicates that the access control point is only available with a PCIXCC/CEX2C.
2. ** indicates that the access control point is only available with a PCIXCC/CEX2C and requires Requires May 2004 or later version of Licensed Internal Code (LIC).
3. To use PKA Key Generate - Clear or PKA Key Generate - Clone, the PKA Key Generate access control point must be enabled or the callable service will fail.
4. To use SET Block Decompose - PIN ext IPINENC or PIN ext OPINENC, the SET Block Decompose access control point must be enabled or the callable service will fail.
5. Diversified Key Generate - single length or same halves requires either Diversified Key Generate - TDES-ENC or Diversified Key Generate - TDES-DEC be enabled.
6. In order to use ATM Remote Key Loading, TKE users will have to enable the access control points for these functions:
   - Trusted Block Create - Activate an Inactive Trusted Key Block
   - Trusted Block Create - Create Trusted Key Block in Inactive Form
   - PKA Key Import - Import an External Trusted Key Block to internal form
   - Remote Key Export - Generate or export a key for use by a non-CCA node

Chapter 7. Key Token Formats

PKA Key Token Formats

RSA Key Token Formats

RSA Private External Key Token

An RSA private external key token contains the following sections:

- A required PKA token header starting with the token identifier X'1E'
- A required RSA private key section starting with one of the following section identifiers:
  - X'02' which indicates a modulus-exponent form RSA private key section (not optimized) with modulus length of up to 1024 bits for use with the Cryptographic Coprocessor Feature or the PCI Cryptographic Coprocessor.
  - X'08' which indicates an optimized Chinese Remainder Theorem form private key section with modulus bit length of up to 4096 bits for use with the PCICC, PCIXCC or CEX2C.
- A required RSA public key section, starting with the section identifier X'04'
- An optional private key name section, starting with the section identifier X'10'

Table 11 presents the basic record format of an RSA private external key token. All length fields are in binary. All binary fields (exponents, lengths, and so on) are stored with the high-order byte first (left, low-address, S/390 format). All binary fields (exponents, modulus, and so on) in the private sections of tokens are right-justified and padded with zeros to the left.

Table 11. RSA Private External Key Token Basic Record Format

Offset (Dec)	Number of Bytes	Description
Token Header (required)		
000	001	Token identifier. X'1E' indicates an external token. The private key is either in cleartext or enciphered with a transport key-encrypting key.
001	001	Version, X'00'.
002	002	Length of the key token structure.
004	004	Ignored. Should be zero.
RSA Private Key Section (required)		
• For 1024-bit Modulus-Exponent form refer to "RSA Private Key Token, 1024-bit Modulus-Exponent External Form" on page 34 • For 4096-bit Modulus-Exponent form refer to "RSA Private Key Token, 4096-bit Modulus-Exponent External Form" on page 35 • For 4096-bit Chinese Remainder Theorem form refer to "RSA Private Key Token, 4096-bit Chinese Remainder Theorem External Form" on page 36		
RSA Public Key Section (required)		
000	001	X'04', section identifier, RSA public key.
001	001	X'00', version.
002	002	Section length, 12+xxx.
004	002	Reserved field.
006	002	RSA public key exponent field length in bytes, "xxx".

Table 11. RSA Private External Key Token Basic Record Format (continued)

Offset (Dec)	Number of Bytes	Description
008	002	Public key modulus length in bits.
010	002	RSA public key modulus field length in bytes, which is zero for a private token. **Note:** In an RSA private key token, this field should be zero. The RSA private key section contains the modulus.
012	xxx	Public key exponent, e (this is generally a 1-, 3-, or 64- to 512-byte quantity). e must be odd and 1<e<n. (Frequently, the value of e is 2^{16}+1 (=65,537).
Private Key Name (optional)		
000	001	X'10', section identifier, private key name.
001	001	X'00', version.
002	002	Section length, X'0044' (68 decimal).
004	064	Private key name (in ASCII), left-justified, padded with space characters (X'20'). An access control system can use the private key name to verify that the calling application is entitled to use the key.

RSA Private Key Token, 1024-bit Modulus-Exponent External Form: This RSA private key token and the external X'02' token is supported on the Cryptographic Coprocessor Feature and PCI Cryptographic Coprocessor.

Table 12. RSA Private Key Token, 1024-bit Modulus-Exponent External Format

Offset (Dec)	Number of Bytes	Description
000	001	X'02', section identifier, RSA private key, modulus-exponent format (RSA-PRIV)
001	001	X'00', version.
002	002	Length of the RSA private key section X'016C' (364 decimal).
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the section end. This hash value is checked after an enciphered private key is deciphered for use.
024	004	Reserved; set to binary zero.
028	001	Key format and security: **X'00'** Unencrypted RSA private key subsection identifier. **X'82'** Encrypted RSA private key subsection identifier.
029	001	Reserved, binary zero.
030	020	SHA-1 hash of the optional key-name section. If there is no key-name section, then 20 bytes of X'00'.
050	004	Key use flag bits. **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. **6** The key is translatable All other bits reserved, set to binary zero.
054	006	Reserved; set to binary zero.
060	024	Reserved; set to binary zero.
084	Start of the optionally-encrypted secure subsection.	

Table 12. RSA Private Key Token, 1024-bit Modulus-Exponent External Format (continued)

Offset (Dec)	Number of Bytes	Description
084	024	Random number, confounder.
108	128	Private-key exponent, d. $d=e^{-1} \bmod((p-1)(q-1))$, and $1<d<n$ where e is the public exponent.
	End of the optionally-encrypted subsection; the confounder field and the private-key exponent field are enciphered for key confidentiality when the key format and security flags (offset 28) indicate that the private key is enciphered. They are enciphered under a double-length transport key using the ede2 algorithm.	
236	128	Modulus, n. n=pq where p and q are prime and $1<n<2^{1024}$.

RSA Private Key Token, 4096-bit Modulus-Exponent External Form: This RSA private key token and the external X'09' token is supported on the Crypto Express2 Coprocessor.

Table 13. RSA Private Key Token, 4096-bit Modulus-Exponent External Format

Offset (Dec)	Number of Bytes	Description
000	001	X'09', section identifier, RSA private key, modulus-exponent format (RSAMEVAR).
001	001	X'00', version.
002	002	Length of the RSA private key section 132+ddd+nnn+xxx.
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the section end. This hash value is checked after an enciphered private key is deciphered for use.
024	002	Length of the encrypted private key section 8+ddd+xxx.
026	002	Reserved; set to binary zero.
028	001	Key format and security: **X'00'** Unencrypted RSA private key subsection identifier. **X'82'** Encrypted RSA private key subsection identifier.
029	001	Reserved, set to binary zero.
030	020	SHA-1 hash of the optional key-name section. If there is no key-name section, then 20 bytes of X'00'.
050	001	Key use flag bits. **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. **6** The key is translatable All other bits reserved, set to binary zero.
051	001	Reserved; set to binary zero.
052	048	Reserved; set to binary zero.
100	016	Reserved; set to binary zero.
116	002	Length of private exponent, d, in bytes: ddd.
118	002	Length of modulus, n, in bytes: nnn.
120	002	Length of padding field, in bytes: xxx.
122	002	Reserved; set to binary zero.
124	Start of the optionally-encrypted secure subsection.	

Table 13. RSA Private Key Token, 4096-bit Modulus-Exponent External Format (continued)

Offset (Dec)	Number of Bytes	Description
124	008	Random number, confounder.
132	ddd	Private-key exponent, d. $d=e^{-1} \bmod((p-1)(q-1))$, and $1<d<n$ where e is the public exponent.
132+ddd	xxx	X'00' padding of length xxx bytes such that the length from the start of the random number above to the end of the padding field is a multiple of eight bytes.
	End of the optionally-encrypted subsection; the confounder field and the private-key exponent field are enciphered for key confidentiality when the key format and security flags (offset 28) indicate that the private key is enciphered. They are enciphered under a double-length transport key using the ede2 algorithm.	
132+ddd+xxx	nnn	Modulus, n. n=pq where p and q are prime and $1<n<2^{4096}$.

RSA Private Key Token, 4096-bit Chinese Remainder Theorem External Form: This RSA private key token (up to 2048-bit modulus) is supported on the PCICC, PCIXCC or CEX2C. The 4096-bit modulus private key token is supported on the z9 EC, z9 BC, z10 EC and z10 BC with the Nov. 2007 or later version of the licensed internal code installed on the CEX2C.

Table 14. RSA Private Key Token, 4096-bit Chinese Remainder Theorem External Format

Offset (Dec)	Number of Bytes	Description
000	001	X'08', section identifier, RSA private key, CRT format (RSA-CRT)
001	001	X'00', version.
002	002	Length of the RSA private-key section, 132 + ppp + qqq + rrr + sss + uuu + xxx + nnn.
004	020	SHA-1 hash value of the private key subsection cleartext, offset 28 to the end of the modulus.
024	004	Reserved; set to binary zero.
028	001	Key format and security: **X'40'** Unencrypted RSA private-key subsection identifier, Chinese Remainder form. **X'42'** Encrypted RSA private-key subsection identifier, Chinese Remainder form.
029	001	Reserved; set to binary zero.
030	020	SHA-1 hash of the optional key-name section and any following optional sections. If there are no optional sections, then 20 bytes of X'00'.
050	004	Key use flag bits. **Bit** **Meaning When Set On** **0** Key management usage permitted. **1** Signature usage not permitted. **6** The key is translatable All other bits reserved, set to binary zero.
054	002	Length of prime number, p, in bytes: ppp.
056	002	Length of prime number, q, in bytes: qqq.
058	002	Length of d_p, in bytes: rrr.
060	002	Length of d_q, in bytes: sss.

Table 14. RSA Private Key Token, 4096-bit Chinese Remainder Theorem External Format (continued)

Offset (Dec)	Number of Bytes	Description
062	002	Length of U, in bytes: uuu.
064	002	Length of modulus, n, in bytes: nnn.
066	004	Reserved; set to binary zero.
070	002	Length of padding field, in bytes: xxx.
072	004	Reserved, set to binary zero.
076	016	Reserved, set to binary zero.
092	032	Reserved; set to binary zero.
124	Start of the optionally-encrypted secure subsection.	
124	008	Random number, confounder.
132	ppp	Prime number, p.
132 + ppp	qqq	Prime number, q
132 + ppp + qqq	rrr	$d_p = d \bmod (p - 1)$
132 + ppp + qqq + rrr	sss	$d_q = d \bmod (q - 1)$
132 + ppp + qqq + rrr + sss	uuu	$U = q^{-1} \bmod (p)$.
132 + ppp + qqq + rrr + sss + uuu	xxx	X'00' padding of length xxx bytes such that the length from the start of the random number above to the end of the padding field is a multiple of eight bytes.
	End of the optionally-encrypted secure subsection; all of the fields starting with the confounder field and ending with the variable length pad field are enciphered for key confidentiality when the key format-and-security flags (offset 28) indicate that the private key is enciphered. They are enciphered under a double-length transport key using the TDES (CBC outer chaining) algorithm.	
132 + ppp + qqq + rrr + sss + uuu + xxx	nnn	Modulus, n. n = pq where p and q are prime and $1<n<2^{4096}$.

Chapter 8. Callable services affected by key store policy

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

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

Table 15. Callable services and parameters affected by key store policy

ICSF callable service	31-bit name	Parameter checked
ANSI X9.17 key export	CSNAKEX	source_data_key_1_identifier source_data_key_2_identifier source_key_encrypting_key_identifier transport_key_identifier
ANSI X9.17 key import	CSNAKIM	transport_key_identifier
ANSI X9.17 key translate	CSNAKTR	inbound_transport_key_identifier outbound_transport_key_identifier
ANSI X9.17 transport key	CSNATKN	source_transport_key_identifier
Cipher text translate	CSNBCTT	key_identifier_in key_identifier_out
Clear PIN encrypt	CSNBCPE	PIN_encrypting_key_identifier
Clear PIN generate alternate	CSNBCPA	PIN_encryption_key_identifier PIN_generation_key_identifier
Clear PIN generate	CSNBPGN	PIN_generation_key_identifier
Control vector translate	CSNBCVT	KEK_key_identifier source_key_token array_key_left array_key_right
Cryptographic variable encipher	CSNBCVE	c_variable_encrypting_key_identifier
Data key export	CSNBDKX	source_key_identifier exporter_key_identifier
Data key import	CSNBDKM	source_key_token importer_key_identifier
Decipher	CSNBDEC	key_identifier
Digital signature generate	CSNDDSG	PKA_private_key_identifier

Table 15. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
Digital signature verify	CSNDDSV	PKA_public_key_identifier
Diversified key generate	CSNBDKG	generating_key_identifier generated_key_identifier
Encipher	CSNBENC	key_identifier
Encrypted PIN generate	CSNBEPG	PIN_generating_key_identifier outbound_PIN_encrypting_key_ identifier
Encrypted PIN translate	CSNBPTR	input_PIN_encrypting_key_identifier output_PIN_encrypting_key_identifier
Encrypted PIN verify	CSNBPVR	input_PIN_encrypting_key_identifier PIN_verifying_key_identifier
Key export	CSNBKEX	source_key_identifier exporter_key_identifier
Key generate	CSNBKGN	KEK_key_identifier_1 KEK_key_identifier_2
Key import	CSNBKIM	source_key_token importer_key_identifier
Key test	CSNBKYT	key_identifier
Key test extended	CSNBYTX	key_identifier kek_key_identifier
Key translate	CSNBKTR	input_key_token input_KEK_key_identifier
MAC generate	CSNBMGN	key_identifier
MAC verify	CSNBMGN	key_identifier
Multiple secure key import	CSNBSKM	key_encrypting_key_identifier
PIN Change/Unblock	CSNBPCU	authentication_issuer_master_key_ identifier encryption_issuer_master_key_identifier new_reference_PIN_key_identifier current_reference_PIN_key_identifier
PKA decrypt	CSNDPKD	PKA_key_identifier
PKA encrypt	CSNDPKE	PKA_key_identifier
PKA key generate	CSNDPKG	transport_key_identifier
PKA key import	CSNDPKI	exporter_key_identifier

Table 15. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
PKA key translate	CSNDPKT	source_key_identifier source_transport_key_identifier target_transport_key_identifier
PKA key token change	CSNDPKTC	key_identifier
PKA public key extract	CSNDPKX	source_key_identifier target_public_key_token
Prohibit export	CSNBPEX	key_identifier
Prohibit export extended	CSNBPEXX	source_key_token, kek_key_identifier
Remote key export	CSNDRKX	trusted_block_identifier transport_key_identifier importer_key_identifier source_key_identifier
Secure key import	CSNBSKI	importer_key_identifier key_identifier
Secure messaging for keys	CSNBSKY	input_key_identifier key_encrypting_key_identifier secmsg_key_identifier
Secure messaging for PINs	CSNBSPN	PIN_encrypting_key_identifier secmsg_key_identifier
SET block compose	CSNDSBC	RSA_public_key_identifier DES_key_block RSA_OAEP_block
SET block decompose	CSNDSBD	RSA_private_key_identifier DES_key_block (one or two tokens)
Symmetric algorithm decipher	CSNBSAD	key_identifier
Symmetric algorithm encipher	CSNBSAE	key_identifier
Symmetric key decipher	CSNBSYD	key_identifier
Symmetric algorithm encipher	CSNBSYE	key_identifier
Symmetric key export	CSNDSYX	DATA_key_identifier RSA_public_key_identifier

Table 15. Callable services and parameters affected by key store policy (continued)

ICSF callable service	31-bit name	Parameter checked
Symmetric key generate	CSFSYG	key_encrypting_key_identifier RSA_public_key_identifier DES_enciphered_key_token
Symmetric key import	CSNDSYI	RSA_enciphered_key RSA_private_key_identifier
Transaction validation	CSNBTRV	transaction_key_identifier
Transform CDMF key	CSNBTCK	source_key_identifier kek_key_identifier
Trusted block create	CSNDTBC	input_block_identifier transport_key_identifier
User derived key	CSFUDK	derivation_key_identifier source_key_identifier
VISA CVV service generate	CSNBCSG	CVV_key_A_Identifier CVV_key_B_Identifier
VISA CVV service verify	CSNBCSV	CVV_key_A_Identifier CVV_key_B_Identifier

The callable services that are affected by the no duplicates key store policy controls are listed in the table below.

Table 16. Callable services that are affected by the no duplicates key store policy controls

ICSF callable service	31-bit name	Parameter checked
Key part import	CSNBKPI	key_identifier
Key record write	CSNBKRW	key_token
PKA Key Generate	CSNDPKG/CSNFPKG	generated_key_token
PKA Key Import	CSNDPKI/CSNFPKI	source_key_identifier
PKDS record create	CSNDKRC/CSNFKRC	token
PKDS record read	CSNDKRR	token
PKDS record write	CSNDKRW	key_token
Trusted Block Create	CSNDTBC	input_block_identifier

Summary of Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions

For services that are passed a label, the key store policy will not affect the SAF check, so only Granular Keylabel Access Controls and CSNDSYX Access Controls will have an effect:

Table 17. Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions (label)

	No CSNDSYX Access Controls for algorithm	**CSNDSYX Access Controls for algorithm**	**No Granular Keylabel Access Controls**	**Granular Keylabel Access Controls**
CSNDSYX: DATA key identifier	label SAF check is done against CSFKEYS	label SAF check is done against XCSFKEY	n/a	n/a
CSNDSYX: RSA key identifier and all other services passed a label	n/a	n/a	label SAF check is done against CSFKEYS for READ access	label SAF check is done against CSFKEYS for appropriate access

For services that are passed a token:

Table 18. Key Store Policy (KSP) and Enhanced Keylabel Access Control interactions (token)

	No KSP	**KSP**			
		No CSNDSYX Access Controls for algorithm	**CSNDSYX Access Controls for algorithm**	**No Granular Keylabel Access Controls**	**Granular Keylabel Access Controls**
CSNDSYX: DATA key identifier	no SAF check is done	KSP SAF checks are done against CSFKEYS	KSP SAF checks are done against XCSFKEY	n/a	n/a
CSNDSYX: RSA key identifier and all other services passed a label	no SAF check is done	n/a	n/a	KSP SAF checks are done against CSFKEYS	KSP SAF checks are done against CSFKEYS

Note: The levels used by Granular Keylabel Access Controls will also be applied to KSP checks (that is, if the CKDS labels matching a token were checked with UPDATE access, CSF-CKDS-DEFAULT will also be checked with UPDATE access)