Key Management Framework

API Reference

Revision	*Date*	*Author*
0.5	7/2/2007	W. Ingersoll

Table of Contents

1 Introduction

The Key Management Framework provides programming interfaces for developers who want a unified, supported set of APIs for writing applications that make use of PKI objects such as X.509 certificate RSA keypairs. This document describes the data types used, as well as a description of all of the functions that are publicly available for use by developers.

2 Programming Reference

The following sections describe the various programming interfaces that are exposed by the Key Mangement Framework. Consumers of these interfaces must include */usr/include/kmfapi.h*. This file contains all of the function prototype and also includes the KMF typedefs and structures needed by some of the routines below.

2.1 kmf_configure_keystore

```
KMF_RETURN
kmf_configure_keystore(
        KMF_HANDLE_T          handle,
        int                   num_args,
        KMF_ATTRIBUTE         *attrlist);
```

2.1.1 Description

KMF currently supports PKCS#11 tokens, NSS databases, and OpenSSL files as potential keystores. When using PKCS#11 and NSS, there are some preliminary steps that must be taken to enable access to the desired token or database that will be used. The `kmf_configure_keystore` function is used to prepare the keystore for further operations.

When using PKCS#11, this function will select a token based on the token name given by the caller.

When using NSS, this function is used to identify the NSS database to be used. Information such as the configuration directory name, certificate database prefix, key database prefix and a security module name can be provided.

If the application is only going to use OpenSSL files, this step is **not** necessary.

2.1.2 Supported Attributes:

Table 1 – kmf_configure_keystore Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to configure
KMF_DIRPATH_ATTR	char *	NSS only
KMF_CERTPREFIX_ATTR	char *	NSS only
KMF_KEYPREFIX_ATTR	char *	NSS only
KMF_SECMODNAME_ATTR	char *	NSS only

KMF_TOKEN_LABEL_ATTR	char *	PKCS#11 only
KMF_READONLY_ATTR	boolean_t	PKCS#11 only

2.1.3 Return Codes:

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_UNINITIALIZED
- KMF_ERR_TOKEN_SELECTED
- KMF_ERR_INTERNAL
- KMF_ERR_MEMORY
- KMF_ERR_AUTH_FAILED

2.2 kmf_initialize

```
KMF_RETURN
kmf_initialize(KMF_HANDLE_T       *outhandle,
     char                         *policy_filename,
     char                         *policy_name);
```

2.2.1 Description

kmf_initialize creates an opaque session handle (KMF_HANDLE_T) that is used in almost all subsequent KMF function calls. This function also initializes some internal state that is associated with the handle. The caller may specify an alternate policy database filename (*/etc/security/kmfpolicy.dtd* is used if NULL) or an alternate policy within a database (*default* is used if NULL).

2.2.2 Return Codes

- KMF_OK
- KMF_ERR_MEMORY
- KMF_ERR_UNINITIALIZED
- KMF_OK
- KMF_ERR_UNINITIALIZED
- KMF_ERR_MEMORY
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_POLICY_DB_FORMAT
- KMF_ERR_POLICY_NOT_FOUND
- KMF_ERR_PLUGIN_INIT

2.3 kmf_dn_parser

```
KMF_RETURN
kmf_dn_parser(char *name_string, KMF_X509_NAME *dname);
```

2.3.1 Description

Applications often need to interpret a distinguished name in human-readable format and encode it so that it can be used in a certificate, CRL, or CSR. This function parses a DN in standard form: "*C=US, O=Widget, Inc., CN=someone@foo.com*" . Fields may be separated by a comma or a semi-colon. The following names are recognized:

CN – Common Name

L – Locality

ST – State or Province

O – Organization Name

OU – Organizational Unit Name

C – Country Name

STREET – Street Address

DC – DC

UID – RFC1274 UID

E - PKCS#9 Email Address

MAIL – RFC1274 Mail

2.3.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_RDN_PARSER

2.4 kmf_oid_to_string

```
char *kmf_oid_to_string(KMF_OID *oid);
```

2.4.1 Description

This is a utility routine for converting a KMF_OID value into a human readable form.

2.4.2 Return Values

- This function returns a string representation of the OID

2.5 kmf_string_to_oid

```
KMF_RETURN
kmf_string_to_oid(char *string, KMF_OID *oid);
```

2.5.1 Description

Convert a human-readable OID string to the internal KMF_OID structure.

2.5.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY

2.6 kmf_compare_rdns

```
int kmf_compare_rdns(
   KMF_X509_NAME *name1,
   KMF_X509_NAME *name2);
```

2.6.1 Description

This function is able to compare to KMF_X509_NAME (relative distinguished name) structures by sorting the various DN fields and comparing them individually. This is a more accurate way of comparing them than to compare the human-readable string formats.

2.6.2 Return Codes

- 0 = The RDNs are equal
- 1 = The RDNs differ.

2.7 kmf_string_to_ku

```
uint32_t
kmf_string_to_ku(char *key_usage_string);
```

2.7.1 Description

This function is mostly used to convert one of the following key usage strings to 32-bit integer values:

- digitalSignature
- nonRepudiation
- keyEncipherment
- dataEncipherment
- keyAgreement
- keyCertSign
- cRLSign
- encipherOnly

- decipherOnly

2.7.2 Return Codes

This function returns one of the following KMF constants:

- KMF_digitalSignature (0x8000)
- KMF_nonRepudiation (0x4000)
- KMF_keyEncipherment (0x2000)
- KMF_dataEncipherment (0x1000)
- KMF_keyAgreement (0x0800)
- KMF_keyCertSign (0x0400)
- KMF_cRLSign (0x0200)
- KMF_encipherOnly (0x0100)
- KMF_decipherOnly (0x0080)

2.8 kmf_ku_to_string

```
char *
kmf_ku_to_string(uint32_t key_usage_value);
```

2.8.1 Description

This function takes an integer key usage constant and returns the human-readable string associated with that value.

2.8.2 Return Values

This function will return NULL if the value given does not match one of the allowed key usage values (see 1.7.2 above), otherwise it returns the human-readable string representation.

2.9 kmf_ekuname_to_oid

```
KMF_OID *
kmf_ekuname_to_oid(char *ekuname);
```

2.9.1 Description

This function recognizes a small set of Extended Key Usage strings and returns the appropriate OID value. The set of recognized strings is:

- “serverAuth”
- “clientAuth”
- “codeSigning”
- “emailProtection”
- “ipsecEndSystem”
- “ipsecTunnel”

- “ipsecUser”
- “timeStamping”
- “OCSPSigning”

If one of these is not found, a NULL value is returned.

2.9.2 Return Code

This function will return a pointer to an OID record if a match is found or NULL if not.

2.10 kmf_oid_to_eku_string

```
char *kmf_oid_to_eku_string(KMF_OID *oid);
```

2.10.1 Description

This function takes an OID and tries to map it to one of the common EKU names that KMF understands (see 1.9.1 above).

2.10.2 Return Codes

Return the string (see 1.9.1 above) associated with the OID if found, or NULL if not.

2.11 kmf_set_token_pin

```
KMF_RETURN
kmf_set_token_pin(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist);
```

2.11.1 Description

This function is used to set the PIN for a PKCS#11 token or NSS database.

2.11.2 Supported Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to configure
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL *	Credential record containing the current PIN
KMF_NEWPIN_ATTR	KMF_CREDENTIAL *	Credential record containing the new PIN
KMF_TOKEN_LABEL_ATTR	char *	Label identifying the token that is being changed.
KMF_SLOT_ID_ATTR	CK_SLOT_ID	If the PKCS#11 slot id is already known, it can be used here.

2.11.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_FUNCTION_NOT_FOUND
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_INTERNAL

2.12 kmf_select_token

```
KMF_RETURN
kmf_select_token(
   KMF_HANDLE_T          handle,
   char              *label,
   int               readonly);
```

2.12.1 Description

This function allows an application to select a particular PKCS#11 token to use for internal cryptographic operations. It initializes the cryptoki session handle that is to be linked internally with the KMF handle.

2.12.2 Return Codes

- KMF_OK
- KMF_ERR_TOKEN_SELECTED
- KMF_ERR_INTERNAL

2.13 kmf_get_pk11_handle

```
CK_SESSION_HANDLE
kmf_get_pk11_handle(KMF_HANDLE_T handle)
```

2.13.1 Description

Returns the cryptoki session handle associated with the KMF handle. This is useful for applications that use both PKCS#11 APIs and KMF APIs and need access to the session handle.

2.13.2 Return Codes

Returns the current session handle associated with the KMF handle. If not yet initialized, it will return 0 (CK_INVALID_HANDLE).

2.14 kmf_pk11_token_lookup

```
KMF_RETURN
kmf_pk11_token_lookup(
   KMF_HANDLE_T          handle,
```

```
    char                    *label,
    CK_SLOT_ID        *slot_id);
```

2.14.1 Description

This is a utility function which will search for a PKCS#11 token associated with the "label" given. If found, it will return the proper slot value in the "slot_id" variable. The KMF_HANDLE_T parameter is optional (it may be given as NULL).

2.14.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_UNINITIALIZED
- KMF_ERR_MEMORY
- KMF_ERR_TOKEN_NOT_PRESENT

2.15 kmf_set_csr_pubkey

```
KMF_RETURN
kmf_set_csr_pubkey(
     KMF_HANDLE_T        handle,
     KMF_KEY_HANDLE      *key,
     KMF_CSR_DATA        *CSR);
```

2.15.1 Description

Attach subject public key information to a CSR that has not yet been signed.

2.15.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_UNINITIALIZED
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING

2.16 kmf_set_csr_version

```
KMF_RETURN
kmf_set_csr_version(KMF_CSR_DATA *CSR, uint32_t version);
```

2.16.1 Description

Set the version number for the CSR being created. The version value must be 0, 1 or 2 to represent the X.509 certificate version (v1 = 0, v2 = 1, v3 =2). A value of 2 (v3) is recommended for most applications.

2.16.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY

2.17 kmf_set_csr_subject

```
KMF_RETURN
kmf_set_csr_subject(
   KMF_CSR_DATA          *CSR,
   KMF_X509_NAME         *subject);
```

2.17.1 Description

Set the Subject Name in a CSR Record that has not yet been signed.

2.17.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

2.18 kmf_create_csr_file

```
KMF_RETURN
kmf_create_csr_file(
   KMF_DATA              *csrdata,
   KMF_ENCODE_FORMAT     format,
   char                  *csrfile);
```

2.18.1 Description

Given a block of raw ASN.1 encoded CSR data, generate a CSR file in either PEM or DER format.

2.18.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_OPEN_FILE
- KMF_ERR_WRITE_FILE
- KMF_ERR_MEMORY

2.19 kmf_set_csr_extn

```
KMF_RETURN
kmf_set_csr_extn(
   KMF_CSR_DATA *csrdata,
   KMF_X509_EXTENSION *extn);
```

2.19.1 Description

This is a generic routine for adding any X509v3 extension to a certificate record. The data must be properly ASN.1 encoded in the KMF_X509_EXTENSION record (defined in header file).

2.19.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY

2.20 kmf_set_csr_sig_alg

```
KMF_RETURN
kmf_set_csr_sig_alg(
   KMF_CSR_DATA          *csrdata,
   KMF_ALGORITHM_INDEX   sigalg);
```

2.20.1 Description

Set the Signature Algorithm in a CSR record that has not yet been signed.

2.20.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

2.21 kmf_set_csr_subject_altname

```
KMF_RETURN
kmf_set_csr_subject_altname(
   KMF_CSR_DATA               *csrdata,
   char                       *altname,
   int                        critical,
   KMF_GENERALNAMECHOICES     alttype);
```

2.21.1 Description

Set the SubjectAltName extension in a CSR Record that has not yet been signed.

KMF currently supports the following KMF GeneralName constants for the AltName extensions:

- **GENNAME_RFC822NAME** – namestr is expected to be in the form of an email address (ex: myname@sun.com).
- **GENNAME_DNSNAME** – namestr is expected to be in the form of a DNS hostname (ex: www.sun.com)
- **GENNAME_URI** – namestr is expected to be in the form of a URI(ex: http://www.sun.com).
- **GENNAME_IPADDRESS** – namestr is expected to be in the form of an ip address (ex: 10.0.0.1)
- **GENNAME_REGISTEREDID** – namestr is expected to be in the form of an object id (OID)

(ex: 1.2.840.113549.1.1.4).

- **GENNAME_DIRECTORYNAME** – namestr is expected to be in the form of a directory name, like a subject name or issuer name.(Ex: C=US, ST=California, L=Menlo Park, O=Sun Microsystems Inc., OU=Security, CN=John Smith).

  X.400 Addresses, and EDIPartyNames are not supported.

2.21.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_ENCODING
- KMF_ERR_MEMORY

2.22 kmf_set_csr_ku

```
KMF_RETURN kmf_set_csr_ku(
   KMF_CSR_DATA     *csr,
   int              critical,
   uint16_t         kubits)
```

2.22.1 Description

Set the KeyUsage extension in a CSR Record that has not yet been signed. The caller must identify whether or not this value is critical as well as passing in a bitmask of the desired key usage values. Valid Key Usage values are defined in the libkmf header files, and they may be added together to form a set of values

KMF defines a list of constants that can be combined for use in the 'kubits' field:

- KMF_digitalSignature
- KMF_nonRepudiation
- KMF_keyEncipherment
- KMF_dataEncipherment
- KMF_keyAgreement
- KMF_keyCertSign
- KMF_cRLSign
- KMF_encipherOnly
- KMF_decipherOnly

2.22.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_ENCODING
- KMF_ERR_MEMORY

2.23 kmf_sign_csr

```
KMF_RETURN
kmf_sign_csr(
   KMF_HANDLE_T          handle,
   const KMF_CSR_DATA    *csrdata,
   KMF_KEY_HANDLE        *signkey,
   KMF_DATA              *signed_csr);
```

2.23.1 Description

Sign a CSR using the given key and the algorithm in the CSR record. The result is a DER encoded signed CSR that can then be passed on to a CA.

2.23.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_UNINITIALIZED
- KMF_ERR_MEMORY

2.24 kmf_import_crl

```
KMF_RETURN
kmf_import_crl(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist);
```

2.24.1 Description

Import a Certificate Revocation List and store it in one of the supported keystores.

2.24.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to configure
KMF_TOKEN_LABEL_ATTR	char *	Label identifying the token that is being changed.
KMF_DIRPATH_ATTR	char *	Path to the directory where the CRL is stored.
KMF_CRL_FILENAME_ATTR	char *	CRL filename to be imported.
KMF_CRLCHECK_ATTR	boolean_t	Verify the CRL with the signers public key. If this is true, then the signers certificate must be given with the KMF_CERT_FILENAME_ATTR

KMF_CERT_FILENAME_ATTR	char *	Filename containing the signers X.509 certificate. This is required if the KMF_CRLCHECK_ATTR is TRUE.

2.24.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CRL_FORMAT
- KMF_ERR_INTERNAL
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_BAD_CERTFILE
- KMF_ERR_OPEN_FILE

2.25 kmf_delete_crl

```
KMF_RETURN
kmf_delete_crl(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist);
```

2.25.1 Description

Delete a CRL from a keystore.

2.25.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to configure
KMF_TOKEN_LABEL_ATTR	char *	Label identifying the token that is being changed.
KMF_DIRPATH_ATTR	char *	Path to the directory where the CRL is stored.
KMF_CRL_FILENAME_ATTR	char *	CRL filename to be imported.
KMF_CRL_SUBJECT_ATTR	char *	The subject name associated with the CRL to be deleted.
KMF_CRL_ISSUER_NAME	char *	The issuer name associated with the CRL to be deleted.

2.25.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CRL_FORMAT
- KMF_ERR_INTERNAL
- KMF_ERR_OPEN_FILE

2.26 kmf_list_crl

```
KMF_RETURN
kmf_list_crl(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist);
```

2.26.1 Description

List the contents of a CRL.

2.26.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to configure
KMF_TOKEN_LABEL_ATTR	char *	Label identifying the token that is being changed.
KMF_DIRPATH_ATTR	char *	Path to the directory where the CRL is stored.
KMF_CRL_FILENAME_ATTR	char *	CRL filename to be imported.

2.26.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CRL_FORMAT
- KMF_ERR_INTERNAL
- KMF_ERR_OPEN_FILE

2.27 kmf_find_crl

```
KMF_RETURN
kmf_find_crl(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist);
```

2.27.1 Description

List the contents of a CRL.

2.27.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to configure
KMF_TOKEN_LABEL_ATTR	char *	Label identifying the token that is being changed.
KMF_DIRPATH_ATTR	char *	Path to the directory where the CRL is stored.
KMF_CRL_FILENAME_ATTR	char *	CRL filename to be imported.
KMF_CRL_NAMELIST_ATTR	char **	A pointer to a memory address that will be allocated and filled with the resulting CRL if it is found.
KMF_CRL_COUNT_ATTR	char *	A pointer to an integer value. This value will be filled in with the actual number of CRLs that were found to match.

2.27.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CRL_FORMAT
- KMF_ERR_INTERNAL
- KMF_ERR_OPEN_FILE

2.28 kmf_find_cert_in_crl

```
KMF_RETURN
kmf_find_crl(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist);
```

2.28.1 Description

Check to see if a particular certificate is listed in a CRL.

2.28.2 Attribute Table

Attribute Type	Value Type	Notes

KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to configure
KMF_TOKEN_LABEL_ATTR	char *	Label identifying the token that is being changed.
KMF_DIRPATH_ATTR	char *	Path to the directory where the CRL is stored.
KMF_CRL_FILENAME_ATTR	char *	CRL filename to be imported.
KMF_CERT_LABEL_ATTR	char *	The text label used to identify the certificate. The certificate is found in the keystore based on this label and then checked against the CRL.
KMF_CERT_DATA_ATTR	KMF_DATA *	The actual certificate that is being tested against the CRL.

2.28.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_INTERNAL
- KMF_ERR_ISSUER
- KMF_ERR_EMPTY_CRL
- KMF_ERR_NOT_REVOKED

2.29 kmf_verify_crl_file

```
KMF_RETURN
kmf_verify_crl_file(
   KMF_HANDLE_T          handle,
   char                  *crlfile,
   KMF_DATA              *ta_cert);
```

2.29.1 Description

Verify that a CRL file can be decoded and is currently valid according to the timestamps in the CRL itself. 'crlfile' is the filename for the CRL. 'ta_cert' is the raw ASN.1 X.509v3 trust anchor certificate data.

2.29.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_FUNCTION_NOT_FOUND
- KMF_ERR_INTERNAL
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CRL_FORMAT

- KMF_ERR_BAD_CERT_FORMAT

2.30 kmf_check_crl_date

```
KMF_RETURN
kmf_check_crl_date(
   KMF_HANDLE_T          handle,
   char             *crl_filename);
```

2.30.1 Description

Verify that a CRL is being used within its indicated lifetime (between the last update and next update timestamps in the CRL itself).

2.30.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_FUNCTION_NOT_FOUND
- KMF_ERR_INTERNAL
- KMF_ERR_ENCODING
- KMF_ERR_VALIDITY_PERIOD

2.31 kmf_is_crl_file

```
KMF_RETURN
kmf_is_crl_file(
   KMF_HANDLE_T          handle,
   char             *crl_filename,
   KMF_ENCODE_FORMAT     *pformat);
```

2.31.1 Description

Determine if a file contains valid CRL data and return the format in which it is encoded.

2.31.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_FUNCTION_NOT_FOUND
- KMF_ERR_OPEN_FILE
- KMF_ERR_BAD_CRLFILE

2.32 kmf_check_cert_date

```
KMF_RETURN
kmf_check_cert_date(
   KMF_HANDLE_T          handle,
   KMF_DATA              *cert);
```

2.32.1 Description

Check the validity period of a DER encoded certificate. If there is a validity period adjustment associated with the current policy that is tied to the KMF handle used in this call, then the adjustment factor is used when checking the validity period. The adjustment period is specified as the "validity_adjusttime" parameter when creating or modifying policies with the kmfcfg utility and is applied to both the notBefore and NotAfter dates in the certificate.

For example, If the *validity_adjusttime* if specified as "1h" (1 hour), then the dates in the certificate are adjusted (for the check only) as follows:

```
Cert.notBefore = Cert.notBefore – 1 hour;
Cert.notAfter = Cert.notAfter + 1 hour;

if ( now >= (Cert.notBefore – 1 hour)) &&
   (now <= (Cert.notAfter + 1 hour))
    return KMF_OK;
```

2.32.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_VALIDITY_PERIOD

2.33 kmf_create_cert_file

```
KMF_RETURN
kmf_create_cert_file(
   KMF_DATA            *certdata,
   KMF_ENCODE_FORMAT   format,
   char                *outfile)
```

2.33.1 Description

Export a block of ASN.1 X.509v3 certificate data to a file in either DER or PEM format.

2.33.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CRL_FORMAT
- KMF_ERR_INTERNAL
- KMF_ERR_BAD_CERT_FORMAT

2.34 kmf_decrypt

```
KMF_RETURN
kmf_decrypt(
   KMF_HANDLE_T   handle,
   int             num_attrs,
   KMF_ATTRIBUTE   *attrlist)
```

2.34.1 Description

Decrypt a block of data using the private key associated with the given certificate. If using OpenSSL files, a file containing the private key must be provided. Otherwise, this function finds the private key associated with an input certificate in the specific keystore. Then it uses the private key to decrypt the input ciphertext into an output plaintext buffer. If the KeyUsage extension is set in the input certificate, then the dataEncipherment KeyUsage bit must be set in order for this operation to proceed.

2.34.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to find the private key
KMF_CERT_DATA_ATTR	KMF_DATA *	The DER-encoded certificate data
KMF_CIPHERTEXT_DATA_ATTR	KMF_DATA *	The input cipher text
KMF_PLAINTEXT_DATA_ATTR	KMF_DATA *	The output plain text
KMF_TOKEN_LABEL_ATTR	char *	Optional: NSS only
KMF_DIRPATH_ATTR	char *	Optional: OpenSSL only
KMF_KEY_FILENAME_ATTR	char *	Optional: OpenSSL only

2.34.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_SLOTNAME
- KMF_ERR_BAD_ALGORITHM
- KMF_ERR_BAD_KEYUSAGE

- KMF_ERR_KEY_NOT_FOUND

2.35 kmf_delete_cert_from_keystore

```
KMF_RETURN
kmf_delete_cert_from_keystore(
   KMF_HANDLE_T   handle,
   int            num_attrs,
   KMF_ATTRIBUTE  *attrlist)
```

2.35.1 Description

This function delete one or more certificates in the specific keystore, using the searching optional attributes as the deletion criteria.

2.35.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore
KMF_ISSUER_NAME_ATTR	char *	Optional
KMF_SUBJECT_NAME_ATTR	char *	Optional
KMF_CERT_SERIAL_ATTR	KMF_BIGINT *	Optional
KMF_CERT_VALIDITY_ATTR	KMF_CERT_VALIDITY	Optional
KMF_CERT_LABEL_ATTR	char *	Optional; NSS and PKCS#11 only
KMF_PRIVATE_BOOL_ATTR	boolean_t	Optional; PKCS#11 only; default is FALSE.
KMF_TOKEN_LABEL_ATTR	char *	Optional; NSS only
KMF_DIRPATH_ATTR	char *	OpenSSL only. If the KMF_CERT_FILENAME_ATTR attriburte is not specified, then this function will search all certificate files in this diretory.
KMF_CERT_FILENAME_ATTR	char *	OpenSSL only.

2.35.3 Return Codes

- KMF_OK
- KMF_ERR_UNINITIALIZED
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_CERT_NOT_FOUND
- KMF_ERR_SLOTNAME
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_MEMORY

2.36 kmf_encode_cert_record

```
KMF_RETURN
kmf_encode_cert_record(
     KMF_X509_CERTIFICATE     *cert,
     KMF_DATA                 *encodedCert)
```

2.36.1 Description

Take a KMF certificate structure (KMF_X509_CERTIFICATE) and convert it to a DER encoded block of data. The KMF_X509_CERTIFICATE record must have all of the required elements present in order for this to proceed. The required elements are:

- Version
- Serial Number
- signature algorithm type
- Subject Public Key Information
- Subject Name
- Issuer Name

2.36.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_INCOMPLETE_TBS_CERT

2.37 kmf_encrypt

```
KMF_RETURN
kmf_encrypt(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist)
```

2.37.1 Description

Encrypt a block of data (the KMF_PLAINTEXT_DATA_ATTR) using the public key from a certificate (KMF_CERT_DATA_ATTR) and store the results in a ciphertext buffer (KMF_CIPHERTEXT_DATA_ATTR). If the certificate has the KeyUsage extension, then the dataEncipherment bit must be set.

2.37.2 Attributes

Attribute Type	Value Type	Notes
KMF_CERT_DATA_ATTR	KMF_DATA *	Data buffer containing the certificate to use as the public key for the encryption operation.
KMF_PLAINTEXT_DATA	KMF_DATA *	Plaintext data to be encrypted.
KMF_CIPHERTEXT_DATA	KMF_DATA *	Ouput buffer for the ciphertext. The caller must make sure the buffer is big enough to hold the ciphertext result.

2.37.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND

2.38 kmf_export_pk12

```
KMF_RETURN
kmf_export_pk12(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist)
```

2.38.1 Description

This function will export a certificate, its private key, and any associated CA certs and export them into a file in PKCS#12 format. The caller must provide a passphrase (KMF_PK12CRED_ATTR) to use for protecting the private data in the PKCS#12 file.

2.38.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to store
KMF_OUTPUT_FILENAME_ATTR	char *	Filename of PKCS#12 file to create.
KMF_PK12CRED_ATTR	KMF_CREDENTIAL *	Passphrase to use for protecting the private data in the final pkcs12 file.
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL *	Optional: PIN needed to access private key data when using NSS or PKCS#11 keystores.
KMF_CERT_LABEL_ATTR	char *	Optional: label to use for locating the certificate to be exported
KMF_ISSUER_NAME_ATTR	char *	Issuer name to be used for locating the certificate to be exported.
KMF_SUBJECT_NAME_ATTR	char *	Subject name to be used for

		locating the certificate to be exported.
KMF_BIGINT_ATTR	KMF_BIGINT *	Serial number to use for locating the certificate to be exported.
KMF_DIRPATH_ATTR	char *	Pathname to the directory where certs and keys are located (OpenSSL keystore only)
KMF_CERT_FILENAME_ATTR	char *	Filename containing the certificate to be exported (OpenSSL keystore only).
KMF_KEY_FILENAME_ATTR	char *	Filename containing the private key to be exported (OpenSSL keystore only).

2.38.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_OPEN_FILE
- KMF_ERR_KEY_NOT_FOUND
- KMF_ERR_AMBIGUOUS_PATHNAME
- KMF_ERR_CERT_NOT_FOUND
- KMF_ERR_INTERNAL
- KMF_ERR_ENCODING
- KMF_ERR_MEMORY

2.39 kmf_find_cert

```
KMF_RETURN
kmf_import_cert(
   KMF_HANDLE_T   handle,
   int             num_attrs,
   KMF_ATTRIBUTE  *attrlist)
```

2.39.1 Description

This function searches the specific keystore to find certificates that match the searching optional attributes, then returns the certificates and the number of certificates to the caller. This function returns only the number of certificates that match the criteria, if the output certificate list attribute is not provided.

2.39.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to search

KMF__COUNT_ATTR	int	The number of certifiicates to be returned
KMF_X509_DER_CERT_ATTR	KMF_X509_DER_CERT *	Optional; The certificate list to be returned.
KMF_ISSUER_NAME_ATTR	char *	Optional
KMF_SUBJECT_NAME_ATTR	char *	Optional
KMF_CERT_SERIAL_ATTR	KMF_BIGINT *	Optional
KMF_CERT_VALIDITY_ATTR	KMF_CERT_VALIDITY	Optional
KMF_CERT_LABEL_ATTR	char *	Optional; NSS and PKCS#11 only
KMF_PRIVATE_BOOL_ATTR	boolean_t	Optional; PKCS#11 only; default is FALSE.
KMF_TOKEN_LABEL_ATTR	char *	Optional; NSS only
KMF_DIRPATH_ATTR	char *	OpenSSL only. If the KMF_CERT_FILENAME_ATTR attriburte is not specified, then this function will search all certificate files in this diretory.
KMF_CERT_FILENAME_ATTR	char *	OpenSSL only.

2.39.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_INTERNAL
- KMF_ERR_SLOTNAME
- KMF_ERR_CERT_NOT_FOUND
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_NO_TOKEN_SELECTED

2.40 kmf_find_prikey_by_cert

```
KMF_RETURN
kmf_find_prikey_by_cert(
   KMF_HANDLE_T   handle,
   int            num_attrs,
   KMF_ATTRIBUTE  *attrlist)
```

2.40.1 Description

This function searches the specific keystore to find the private key that matches with the public key in the certificate. For the OpenSSL keystore, a key file should be provided.

2.40.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to search
KMF_CERT_DATA_ATTR	KMF_DATA *	The input DER-encoded certificate data
KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE *	The output private key handle
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL *	NSS and PKCS#11 only
KMF_ENCODE_FORMAT_ATTR	KMF_ENCODE_FORMAT	Optional; PKCS#11 only; An input attribute to indicate the returned key needs be in raw format.
KMF_KEY_FILENAME_ATTR	char *	OpenSSL only.
KMF_TOKEN_LABEL_ATTR	char *	Optional; NSS only.
KMF_DIRPATH_ATTR	char *	Optional; OpenSSL only.

2.40.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_CERT_NOT_FOUND
- KMF_ERR_MEMORY
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_SLOTNAME
- KMF_ERR_MEMORY
- KMF_ERR_KEY_NOT_FOUND

2.41 kmf_import_cert

```
KMF_RETURN
kmf_import_cert(
   KMF_HANDLE_T   handle,
   int             num_attrs,
   KMF_ATTRIBUTE   *attrlist)
```

2.41.1 Description

This function takes a certificate file as input and store it into the specified keystore. This function will auto-detect the format of the certificate file, convert it to a DER-encoded certificate if it is not DER-encoded originally, then store it into the specfic keystore.

2.41.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to store
KMF_CERT_FILENAME_ATTR	char *	The input certificate file only
KMF_CERT_LABEL_ATTR	char *	Optional; NSS and PKCS11 only
KMF_TRUSTFLAG_ATTR	char *	Optional; NSS only
KMF_TOKEN_LABEL_ATTR	char *	Optional; NSS only

2.41.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_SLOTNAME
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_ENCODING
- KMF_ERR_MEMORY

2.42 kmf_import_objects

```
KMF_RETURN
kmf_import_pk12(
   KMF_HANDLE_T     handle,
   char              *filename,
   KMF_CREDENTIAL   *pk12cred,
   KMF_DATA          **certs,
   int               *ncerts,
   KMF_RAW_KEY_DATA **rawkeys,
   int               *nkeys)
```

2.42.1 Description

This function extracts certificates and their associated private keys from a PKCS#12 PDU file or a PEM file . Currently, KMF supports one set of certificates and one private key.

2.42.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_OPEN_FILE
- KMF_ERR_ENCODING
- KMF_ERR_PKCS12_FORMAT
- KMF_ERR_CERT_NOT_FOUND

- KMF_ERR_MEMORY

2.43 kmf_is_cert_file

```
KMF_RETURN
kmf_is_cert_file(
     KMF_HANDLE_T        handle,
     char                *filename,
     KMF_ENCODE_FORMAT   *format)
```

2.43.1 Description

Attempt to determine if a file is actually an X.509 certificate and in which format (DER or PEM) it is encoded. The actual format is returned in the *format* parameter.

2.43.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOT_FOUND
- KMF_ERR_FUNCTION_NOT_FOUND
- KMF_ERR_OPEN_FILE
- KMF_ERR_BAD_CERTFILE
- KMF_ERR_ENCODING

2.44 kmf_sign_cert

```
KMF_RETURN
kmf_sign_cert(
   KMF_HANDLE_T   handle,
   int             num_attrs,
   KMF_ATTRIBUTE  *attrlist)
```

2.44.1 Description

This function takes a private key handle or a certificate as the input value to use for signing another certificate. If a signer-cert is used, the given keystore will be searched for the private key that matches with the public key in the signer certificate. This function then signs a to-be-signed DER-encoded certificate with the private key, then returns a signed DER-encoded certificate. The keyCertSign KeyUsage bit must be set in the signer cert in order to this operation to proceed. Additionally, the cA bit in the BasicConstraints extension must also be set and marked as critical. For the OpenSSL keystore, a key file should be provided if the key handle is not given.

2.44.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to search
KMF_SIGNER_CERT_DATA_ATTR	KMF_DATA *	Optional: The input DER-encoded signer certificate data
KMF_TBS_CERT_DATA_ATTR	KMF_DATA *	The input to-be-signed DER-encoded certificate data
KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE *	Optional: The handle of the private key to be used for signing the certificate.
KMF_CERT_DATA_ATTR	KMF_DATA *	The output signed and DER-encoded certificate data
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL *	Optional; NSS and PKCS#11 only
KMF_KEY_FILENAME_ATTR	char *	Optional: OpenSSL only.
KMF_TOKEN_LABEL_ATTR	char *	Optional; NSS only.
KMF_DIRPATH_ATTR	char *	Optional; OpenSSL only.

2.44.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_BAD_ALGORITHM
- KMF_ERR_UNINITIALIZED
- KMF_ERR_MEMORY
- KMF_ERR_INTERNAL
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_KEY_USAGE
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_KEY_NOT_FOUND

2.45 kmf_sign_data

```
KMF_RETURN
kmf_sign_data(
   KMF_HANDLE_T   handle,
   int             num_attrs,
   KMF_ATTRIBUTE  *attrlist)
```

2.45.1 Description

This function takes either a private key handle or a certificate to sign an arbitrary block of data. If a certificate is used, the keystore will be searched for a private key that matches with the public key in the signer certificate, signs a block of data with the private key, then returns the signature. If the KeyUsage extension is set in the input certificate, then the digitalSignature KeyUsage bit must be set in order for this operation to proceed. For the OpenSSL keystore, a key handle must be provided if the key handle is not provided.

2.45.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to search
KMF_SIGNER_CERT_DATA_ATTR	KMF_DATA *	Optional: The input DER-encoded signer certificate data
KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE *	Optional: The private key handle to be used for signing the data.
KMF_DATA_ATTR	KMF_DATA *	The input data
KMF_OUT_DATA_ATTR	KMF_DATA *	The output signature
KMF_ALGORITHM_INDEX_ATTR	KMF_ALGORITHM_INDEX	Optional; the algorithm to be used for signing. If this attribute is not provided, then this function will sign with the algorithm in the signer certificate.
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL *	Optional; NSS and PKCS#11 only
KMF_KEY_FILENAME_ATTR	char *	Optional: OpenSSL only.
KMF_TOKEN_LABEL_ATTR	char *	Optional; NSS only.
KMF_DIRPATH_ATTR	char *	Optional; OpenSSL only.

2.45.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_BAD_ALGORITHM
- KMF_ERR_UNINITIALIZED
- KMF_ERR_MEMORY
- KMF_ERR_INTERNAL
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_KEY_USAGE
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_KEY_NOT_FOUND

2.46 kmf_store_cert

```
KMF_RETURN
kmf_store_cert(
   KMF_HANDLE_T   handle,
   int             num_attrs,
   KMF_ATTRIBUTE  *attrlist)
```

2.46.1 Description

This function takes a DER-encoded certificate as input and store it into the specified keystore.

2.46.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to store
KMF_CERT_DATA_ATTR	KMF_DATA *	DER-encoded certficate data
KMF_CERT_FILENAME_ATTR	char *	The file to store it; OpenSSL only
KMF_CERT_LABEL_ATTR	char *	Optional; NSS and PKCS11 only
KMF_TRUSTFLAG_ATTR	char *	Optional; NSS only
KMF_TOKEN_LABEL_ATTR	char *	Optional; NSS only
KMF_DIRPATH_ATTR	char *	Optional; OpenSSL only
KMF_ENCODE_FORMAT_ATTR	KMF_ENCODE_FORMAT	Optional; OpenSSL Only ; default is PEM.

2.46.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_SLOTNAME
- KMF_ERR_INTERNAL
- KMF_ERR_ENCODING
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_MEMORY

2.47 kmf_validate_cert

```
KMF_RETURN
kmf_validate_cert(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attributes);
```

2.47.1 Description

Validate a certificate according to the KMF policy associated with the KMF handle given.

2.47.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to store

2.47.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND

2.48 kmf_verify_cert

```
KMF_RETURN
kmf_verify_cert(
   KMF_HANDLE_T          handle,
   int                   numattr,
   KMF_ATTRIBUTE         *attrlist)
```

2.48.1 Description

2.48.2 Attributes

Attribute Type	Value Type	Notes
KMF_CERT_DATA_ATTR	KMF_DATA *	Buffer containing the encoded certificate data.
KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE *	Optional: key to use for verifying the certificate data.
KMF_SIGNER_CERT_DATA_ATTR	KMF_DATA *	Optional: the signer certificate used to verify the signature on the certificate data.

2.48.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND

2.49 kmf_verify_data

```
KMF_RETURN
kmf_verify_data(
```

```
        KMF_HANDLE_T            handle,
        int                     numattr,
        KMF_ATTRIBUTE           *attrlist)
```

2.49.1 Description

Verify a signature on a data buffer. The caller must specify the signature that is being verified along with the data buffer to be used. The caller must either specify a certificate to use as the public key for verification or supply a handle to the public key itself. If a certificate is used and it has the KeyUsage extension, that extension must indicate the *digitalSignature* field in order to be used for verification.

2.49.2 Attributes

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	The keystore type where the key used for verification is stored
KMF_DATA_ATTR	KMF_DATA *	Data buffer to be verified
KMF_IN_SIGN_ATTR	KMF_DATA *	Buffer containing the signature to be verified.
KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE *	Optional: Public key to use for verification
KMF_SIGNER_CERT_DATA_ATTR	KMF_DATA *	Optional: Certificate to use as the public key for the verification.
KMF_ALGORITHM_INDEX_ATTR	KMF_ALGORITHM_INDEX	Indicates the signature algorithm to be used for verification.

2.49.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND

2.50 kmf_finalize

```
KMF_RETURN
kmf_finalize(KMF_HANDLE_T handle);
```

2.50.1 Description

This function is used to close a KMF session and cleanup memory associated with the handle.

2.50.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

2.51 kmf_get_kmf_error_str

```
KMF_RETURN
kmf_get_kmf_error_str(
   KMF_RETRUN          errcode,
   char                **errmsg);
```

2.51.1 Description

This function takes the specified KMF error code and returns the corresponding error string. The memory is allocated internally for the error string and must be later freed by the caller using kmf_free_str().

2.51.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MISSING_ERRCODE
- KMF_ERR_MEMORY

2.52 kmf_get_plugin_error_str

```
KMF_RETURN
kmf_get_plugin_error_str(
   KMF_HANDLE_T        handle,
   char                **msgstr);
```

2.52.1 Description

This function returns the system error string or the error string specific to a plugin error. The memory is allocated internally for the error string and must be later freed by the caller using kmf_free_str().

2.52.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MISSING_ERRCODE
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_MEMORY

2.53 kmf_read_input_file

```
KMF_RETURN
kmf_read_input_file(
   KMF_HANDLE_T        handle,
   char                *filename,
   KMF_DATA            *pdata);
```

2.53.1 Description

This function is used to open a file and read all of its contents into a KMF_DATA record. The memory is allocated internally and must be later freed by the caller using kmf_free_data().

2.53.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_OPEN_FILE
- KMF_ERR_MEMORY
- KMF_ERR_INTERNAL

2.54 kmf_der_to_pem

```
KMF_RETURN
kmf_der_to_pem(
   KMF_OBJECT_TYPE     type,
   unsigned char       *data,
   int                 len,
   unsigned char       **out,
   int                 outlen);
```

2.54.1 Description

This function converts a buffer of DER encoded bytes to a buffer of PEM encoded data.

2.54.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_BAD_OBJECT_TYPE
- KMF_ERR_MEMORY

2.55 kmf_pem_to_der

```
KMF_RETURN
kmf_pem_to_der(
   unsigned char       *in,
   int                 inlen,
   unsigned char       **out,
   int                 outlen);
```

2.55.1 Description

This function converts a buffer of PEM encoded bytes to a buffer of DER encoded bytes.

2.55.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_ENCODING
- KMF_ERR_MEMORY

2.56 kmf_oid_to_string

```
char *
kmf_oid_to_string(KMF_OID *oid);
```

2.56.1 Description

This function converts a KMF_OID data structure to a human-readable string of the form “1.2.4.5”.

2.56.2 Return Code

This function will return a human-readable string associated with the given KMF_OID record or NULL if it cannot be converted.

2.57 kmf_get_file_format

```
KMF_RETURN
kmf_get_file_format(
   char                *filename,
   KMF_ENCODE_FORMAT   *fmt);
```

2.57.1 Description

This function is used to determine the format of a file. It attempts to determine if a file is a DER encoded file, a PEM encoded file, or PKCS#12 file. If it can not be determined, KMF_ERR is returned as an error.

2.57.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_ENCODING
- KMF_ERR_OPEN_FILE
- KMF_ERR_MEMORY
- KMF_ERR_INTERNAL

2.58 kmf_hexstr_to_bytes

```
KMF_RETURN
kmf_hexstr_to_bytes(
```

```
    unsigned char       *hexstr,
    unsigned char       **bytes,
    size_t              *outlen);
```

2.58.1 Description

This function converts a hexadecimal string into a byte string. The memory is allocated internally for the returned string and must be later freed by the caller using kmf_free_str().

2.58.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_BAD_HEX_STRING
- KMF_ERR_MEMORY

2.59 kmf_free_dn

```
void
kmf_free_dn(KMF_X509_NAME *name);
```

2.59.1 Description

This function frees the memory associated with a KMF_X509_NAME record.

2.60 kmf_free_kmf_cert

```
void
kmf_free_kmf_cert(
   KMF_HANDLE_T        handle,
   KMF_X509_DER_CERT   *kmf_cert);
```

2.60.1 Description

This function frees the memory associated with a KMF_X509_DER_CERT record. A KMF_X509_DER_CERT record is returned by kmf_find_cert() and can be cleaned up with this function.

2.61 kmf_free_data

```
void
kmf_free_data(KMF_DATA *datablock);
```

2.61.1 Description

This function frees the memory associated with a KMF_DATA record.

2.62 kmf_free_algoid

```
void
```

```
kmf_free_algoid(KMF_X509_ALGORITHM_IDENTIFIER *algoid);
```

2.62.1 Description

This function frees the memory associated with a KMF_X509_ALGORITHM_IDENTIFIER data structure.

2.63 kmf_free_extn

```
void
kmf_free_extn(KMF_509_EXTENSION *exptr);
```

2.63.1 Description

This function frees the memory associated with a KMF_X509_EXTENSION data structure.

2.64 kmf_free_tbs_csr

```
void
kmf_free_tbs_csr(KMF_TBS_CSR *tbscsr);
```

2.64.1 Description

This function frees the memory associated with a KMF_TBS_CSR data structure, which is a Certificate Signing Request record that has not yet been signed.

2.65 kmf_free_signed_csr

```
void
kmf_free_signed_csr(KMF_CSR_DATA *csr);
```

2.65.1 Description

This function frees the memory associated with a signed KMF_CSR_DATA record.

2.66 kmf_free_tbs_cert

```
void
kmf_free_tbs_cert(
   MF_X509_TBS_CERT *tbscert);
```

2.66.1 Description

This function frees the memory associated with a KMF_X509_TBS_CERTdata structure.

2.67 kmf_free_signed_cert

```
void
kmf_free_signed_cert(KMF_X509_CERTIFICATE *certptr);
```

2.67.1 Description

This function frees the memory associated with a signed KMF_X509_CERTIFICATE record.

2.68 kmf_free_str

```
void
kmf_free_str(char *pstr);
```

2.68.1 Description

This function frees the character strings returned by routines such as kmf_get_kmf_error_str() or kmf_get_plugin_error_str().

2.69 kmf_free_eku

```
void
mf_free_eku(KMF_X509EXT_EKU *eptr);
```

2.69.1 Description

This function frees the memory associated with a KMF_X509EXT_EKU data structure.

2.70 kmf_free_spki

```
void
kmf_free_spki(KMF_X509_SPKI *spki);
```

2.70.1 Description

This function frees the memory associated with a KMF_X509_SPKI data structure.

2.71 kmf_free_kmf_key

```
void
kmf_free_kmf_key(
   KMF_HANDLE_T        handle,
   KMF_KEY_HANDLE      *key);
```

2.71.1 Description

This function frees the memory associated with a KMF_KEY_HANDLE data structure. It does NOT destroy the key if it is stored in the token or in the NSS database.

2.72 kmf_free_bigint

```
void
kmf_free_bigint(KMF_BIGINT *big);
```

2.72.1 Description

This function frees the memory associated with a KMF_BIGINT record.

2.73 kmf_free_raw_key

```
void
kmf_free_raw_key(KMF_RAW_KEY_DATA *key);
```

2.73.1 Description

This function frees the memory associated with a KMF_RAW_KEY_DATA record.

2.74 kmf_free_raw_sym_key

```
void
kmf_free_raw_sym_key(KMF_RAW_SYM_KEY *key);
```

2.74.1 Description

This function frees the memory associated with a KMF_RAW_SYM_KEY record.

2.75 kmf_free_crl_dist_pts

```
void
kmf_free_crl_dist_pts(KMF_X509EXT_CRLDISTPOINTS *crl_dps);
```

2.75.1 Description

This function frees the memory associated with a KMF_X509EXT_CRLDISTPOINTS record.

2.76 kmf_create_ocsp_request

```
KMF_RETURN
kmf_create_ocsp_request(
   KMF_HANDLE_T        handle,
   int                 numattr,
   KMF_ATTRIBUTE       *attrlist);
```

2.76.1 Description

This function creates an OCSP Request file that can later be used to send to an OCSP responder.

2.76.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_USER_CERT_DATA_ATTR	KMF_DATA *	User certificate data record.
KMF_ISSUER_CERT_DATA_ATTR	KMF_DATA *	Issuer certificate data record.

KMF_OCSP_REQUEST_FILENAME_ATTR	char *	OCSP request filename to be created.

2.76.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_FUNCTION_NOT_FOUND
- KMF_ERR_OCSP_CREATE_REQUEST
- KMF_ERR_OPEN_FILE
- KMF_ERR_ENCODING
- KMF_ERR_OCSP_BAD_ISSUER
- KMF_ERR_OCSP_BAD_CERT
- KMF_ERR_OCSP_CERTID

2.77 kmf_get_ocsp_status_for_cert

```
KMF_RETURN
kmf_get_ocsp_status_for_cert(
   KMF_HANDLE_T        handle,
   int                 numattr,
   KMF_ATTRIBUTE       *attrlist);
```

2.77.1 Description

This function is made available for applications that need to interpret an OCSP response.

2.77.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_USER_CERT_DATA_ATTR	KMF_DATA *	User certificate data record.
KMF_ISSUER_CERT_DATA_ATTR	KMF_DATA *	Issuer certificate data record.
KMF_RESPONSE_DATA_ATTR	KMF_DATA *	OCSP response data record.
KMF_IGNORE_RESPONSE_SIGN_ATTR	boolean_t	Optional. It specifies whether the verification of response signature is required or not.
KMF_SIGNER_CERT_DATA_ATTR	KMF_DATA *	Optional. OCSP responder certificate data record to be used for response signature verification.
KMF_RESPONSE_LIFETIME_ATTR	uint32_t	Optional. It is used to validate if the OCSP response life time is valid.

KMF_OCSP_RESPONSE_STATUS_ATTR	int *	OCSP response status.
KMF_OCSP_RESPONSE_REASON_ATTR	int *	OCSP response reason.
KMF_OCSP_RESPONSE_CERT_STATUS_ATTR	int *	OCSP response certificate status.

2.77.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_MEMORY
- KMF_ERR_OCSP_MALFORMED_RESPONSE
- KMF_ERR_OCSP_RESPONSE_STATUS
- KMF_ERR_OCSP_NO_BASIC_RESPONSE
- KMF_ERR_OCSP_CERTID
- KMF_ERR_OCSP_UNKNOWN_CERT
- KMF_ERR_OCSP_STATUS_TIME_INVALID
- KMF_ERR_OCSP_BAD_SIGNER
- KMF_ERR_OCSP_BAD_ISSUER
- KMF_ERR_INTERNAL
- KMF_ERR_OCSP_RESPONSE_SIGNATURE

2.78 kmf_string_to_oid

```
KMF_RETURN
kmf_string_to_oid(
   char                 *oidstring,
   KMF_OID              *oid);
```

2.78.1 Description

This function converts a string representing an OID to a binary representation of an OID that can be used in other KMF function calls.

2.78.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY

2.79 kmf_find_attr

```
int
kmf_find_attr(
   KMF_ATTR_TYPE       type,
```

```
    KMF_ATTRIBUTE       *attrlist,
    int                 numattr);
```

2.79.1 Description

Search a list of attributes for one that matches the given type. Return an index into the attribute list if found. Otherwise, return -1.

2.79.2 Return Codes

- Index = The index into the attribute list for the given attribute type.
- -1 = Given attribute type is not found.

2.80 kmf_set_attr_at_index

```
void
kmf_set_attr_at_index(
   KMF_ATTRIBUTE       *attrlist,
   int                 index,
   KMF_ATTR_TYPE       type,
   void                *pValue,
   uint32_t            len);
```

2.80.1 Description

Given an already allocated attribute list, insert the given attribute information at a specific index in the list.

2.81 kmf_set_attr

```
KMF_RETURN
kmf_set_attr(
   KMF_ATTRIBUTE       *attrlist,
   int                 numattr,
   KMF_ATTR_TYPE       type,
   void                *pValue,
   uint32_t            len);
```

2.81.1 Description

Find an attribute matching a particular type and set its pValue and length fields to the given values. If the attribute list does not contain the attribute type given, KMF_ERR_A TTR_NOT_FOUND is returned.

2.81.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_ATTR_NOT_FOUND
- KMF_ERR_BUFFER_SIZE

2.82 kmf_get_attr_ptr

```
void *
kmf_get_attr_ptr(
   KMF_ATTR_TYPE       type,
   KMF_ATTRIBUTE       *attrlist,
   int                 numattr);
```

2.82.1 Description

Find a particular attribute in a list and return a pointer to its value.

2.82.2 Return Code

This function will return a pointer to the value of the given attribute type if found, otherwise it will return NULL.

2.83 kmf_get_attr

```
KMF_RETURN
kmf_get_attr(
   KMF_ATTR_TYPE       type,
   KMF_ATTRIBUTE       *attrlist,
   int                 numattr,
   void                *outValue,
   uint32_t            *outlen);
```

2.83.1 Description

Find a particular attribute in a list and return the value and length values. The outValue and outlen may be NULL, if the caller doesn't want their values to be filled in.

2.83.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_BUFFER_SIZE

2.84 kmf_get_string_attr

```
KMF_RETURN
kmf_get_string_attr(
   KMF_ATTR_TYPE       type,
   KMF_ATTRIBUTE       *attrlist,
   int                 numattr,
   char                **outstr);
```

2.84.1 Description

Find a string type attribute, allocate it and return the value to the caller.The memory is allocated internally for the output string and must be later freed by the caller.

2.84.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BUFFER_SIZE

2.85 kmf_create_keypair

```
KMF_RETURN
kmf_create_keypair(
   KMF_HANDLE_T        handle,
   int                 numattr,
   KMF_ATTRIBUTE       *attrlist);
```

2.85.1 Description

This function is used to create a public and private keypair (RSA or DSA) in a particular keystore.

2.85.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to create a keypair
KMF_TOKEN_LABEL_ATTR	char *	NSS only. Optional. Slot label of NSS token.
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL	NSS and PKCS#11 only. Credential record containing the PIN.
KMF_STOREKEY_BOOL_ATTR	boolean_t	Optional. Default is TRUE. Specify if the key will be stored in a keystore.
KMF_KEYALG_ATTR	KMF_KEY_ALG	Optional. The default key type is KMF_RSA.
KMF_KEYLENGTH_ATTR	uint32_t	Optional. The default key length is 1024 bits.
KMF_PUBKEY_HANDLE_ATTR	KMF_KEY_HANDLE	Public key handle.
KMF_PRIVKEY_HANDLE_ATTR	KMF_KEY_HANDLE	Private key handle.
KMF_KEYLABEL_ATTR	char *	NSS and PKCS#11 only. Optional. Specify the key label .
KMF_RSAEXP_ATTR	KMF_BIGINT	Optional. RSA exponent parameter.
KMF_DIRPATH_ATTR	char *	OpenSSL only. Optional. OpenSSL keystore directory.
KMF_KEY_FILENAME_ATTR	char *	OpenSSL only.
KMF_ENCODE_FORMAT_ATTR	KMF_ENCODE_FORMAT	OpenSSL only. Optional. Encoded format of the key.

		Default is PEM format.

2.85.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_PLUGIN_INIT
- KMF_ERR_SLOTNAME
- KMF_ERR_UNINITIALIZED_TOKEN
- KMF_ERR_AUTH_FAILED
- KMF_ERR_MEMORY
- KMF_ERR_KEYGEN_FAILED
- KMF_ERR_UNINITIALIZED
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_INTERNAL

2.86 kmf_delete_key_from_keystore

```
KMF_RETURN
kmf_delete_key_from_keystore(
   KMF_HANDLE_T       handle,
   int                numattr,
   KMF_ATTRIBUTE      *attrlist);
```

2.86.1 Description

This function is used to delete a key from a particular keystore.

2.86.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to delete a key.
KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE	Key handle
KMF_DESTROY_BOOL_ATTR	boolean_t	Optional. Default is TRUE. Specify whether the key will be deleted from the token.
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL	NSS and PKCS#11 only. It is required when KMF_DESTROY_BOOL_ATTR is TRUE.
KMF_TOKEN_LABEL_ATTR	char *	NSS only. Optional. Slot label of NSS token.

2.86.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_PLUGIN_INIT
- KMF_ERR_SLOTNAME
- KMF_ERR_UNINITIALIZED_TOKEN
- KMF_ERR_AUTH_FAILED
- KMF_ERR_KEY_NOT_FOUND
- KMF_ERR_UNINITIALIZED
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_BAD_KEY_CLASS
- KMF_ERR_INTERNAL

2.87 kmf_find_key

```
KMF_RETURN
kmf_find_key(
   KMF_HANDLE_T        handle,
   int                 numattr,
   KMF_ATTRIBUTE       *attrlist);
```

2.87.1 Description

This function is used to search for a key or a list of keys from a particular keystore.

2.87.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to search for key.
KMF_COUNT_ATTR	uint32_t	Number of keys.
KMF_TOKEN_LABEL_ATTR	char *	NSS only. Optional. Slot label of NSS token.
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL	NSS and PKCS#11 only. Optional for PKCS#11 token.
KMF_KEYCLASS_ATTR	KMF_KEY_CLASS	Key class.
KMF_KEYLABEL_ATTR	char *	Key label.
KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE	Optional. Key handle.
KMF_KEYALG_ATTR	KMF_KEY_ALG	Optional.
KMF_TOKEN_BOOL_ATTR	boolean_t	PKCS#11 only. Specify if the key is a token object.
KMF_IDSTR_ATTR	char *	PKCS#11 only. Optional. Key identifier.

KMF_PRIVATE_BOOL_ATTR	boolean_t	PKCS#11 only. Optional. Specify if the key is a private object.
KMF_ENCODE_FORMAT_ATTR	KMF_ENCODE_FORMAT	PKCS#11 only. Optional.
KMF_DIRPATH_ATTR	char *	OpenSSL only. Optional.
KMF_KEY_FILENAME_ATTR	char *	OpenSSL only. Optional.

2.87.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_PLUGIN_INIT
- KMF_ERR_SLOTNAME
- KMF_ERR_UNINITIALIZED_TOKEN
- KMF_ERR_AUTH_FAILED
- KMF_ERR_KEY_NOT_FOUND
- KMF_ERR_UNINITIALIZED
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_BAD_KEY_CLASS
- KMF_ERR_BAD_KEY_TYPE
- KMF_ERR_INTERNAL

2.88 kmf_create_sym_key

```
KMF_RETURN
kmf_create_sym_key(
   KMF_HANDLE_T        handle,
   int                 numattr,
   KMF_ATTRIBUTE       *attrlist);
```

2.88.1 Description

This function creates a symmetric key. KMF supports the creation of the following symmetric key types:

- Generic Secret
- AES
- DES
- RC4

2.88.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Keystore where symmetric key will be created.

KMF_KEY_HANDLE_ATTR	KMF_KEY_HANDLE	Key handle
KMF_KEYALG_ATTR	KMF_KEY_ALG	Symmetric key type.
KMF_KEYLENGTH_ATTR	uint32_t	Optional.
KMF_TOKEN_LABEL_ATTR	char *	NSS only. Optional. Slot label of NSS token.
KMF_KEYLABEL_ATTR	char *	NSS and PKCS#11 only.
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL	NSS and PKCS#11 only.
KMF_SENSITIVE_BOOL_ATTR	boolean_t	PKCS#11 only.
KMF_NON_EXTRACTABLE_BOOL_ATTR	boolean_t	PKCS#11 only.
KMF_DIRPATH_ATTR	char *	OpenSSL only. Optional.
KMF_KEY_FILENAME_ATTR	char *	OpenSSL only.

2.88.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_BAD_KEY_SIZE
- KMF_ERR_BAD_KEY_TYPE
- KMF_ERR_KEYGEN_FAILED
- KMF_ERR_PLUGIN_INIT
- KMF_ERR_SLOTNAME
- KMF_ERR_UNINITIALIZED_TOKEN
- KMF_ERR_AUTH_FAILED
- KMF_ERR_UNINITIALIZED
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_BAD_KEY_TYPE
- KMF_ERR_BAD_KEY_SIZE
- KMF_ERR_DUPLICATE_KEYFILE
- KMF_ERR_OPEN_FILE
- KMF_ERR_MEMORY

2.89 kmf_get_sym_key_value

```
KMF_RETURN
kmf_get_sym_key_value(
   KMF_HANDLE_T        handle,
   KMF_KEY_HANDLE      *symkey,
   KMF_RAW_SYM_KEY     *rkey);
```

2.89.1 Description

This function returns the raw symmetric key data bytes for a given KMF key handle.

2.89.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_UNINITIALIZED
- KMF_ERR_BAD_KEY_CLASS
- KMF_ERR_BAD_KEYHANDLE
- KMF_ERR_MEMORY
- KMF_ERR_GETKEYVALUE_FAILED
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_OPEN_FILE
- KMF_ERR_INTERNAL

2.90 kmf_store_key

```
KMF_RETURN
kmf_store_key(
   KMF_HANDLE_T        handle,
   int                 numattr,
   KMF_ATTRIBUTE       *attrlist);
```

2.90.1 Description

This function stores a key into the keystore. The key can be a public key, private key or a raw key data.

2.90.2 Attribute Table

Attribute Type	Value Type	Notes
KMF_KEYSTORE_TYPE_ATTR	KMF_KEYSTORE_TYPE	Which keystore to search for key.
KMF_TOKEN_LABEL_ATTR ok	char *	NSS only. Optional. Slot label of NSS token.
KMF_CREDENTIAL_ATTR	KMF_CREDENTIAL *	Required for NSS and PKCS#11. Optional for OpenSSL.
KMF_PUBKEY_HANDLE_ATTR	KMF_KEY_HANDLE *	Public key to be stored.
KMF_PRIVKEY_HANDLE_ATTR	KMF_KEY_HANDLE *	Private key to be stored.
KMF_RAW_KEY_ATTR	KMF_RAW_KEY_DATA *	Raw key data to be stored.
KMF_KEYLABEL_ATTR	char *	Optional.
KMF_CERT_DATA_ATTR	KMF_DATA *	NSS and PKCS#11 only. It is required when storing a raw key data.
KMF_DIRPATH_ATTR	char *	OpenSSL only. Optional.
KMF_KEY_FILENAME_ATTR	char *	OpenSSL only.

KMF_ENCODE_FORMAT_ATTR		OpenSSL only. Optional.

2.90.3 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_PLUGIN_INIT
- KMF_ERR_SLOTNAME
- KMF_ERR_UNINITIALIZED_TOKEN
- KMF_ERR_AUTH_FAILED
- KMF_ERR_ATTR_NOT_FOUND
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_INTERNAL
- KMF_ERR_UNINITIALIZED
- KMF_ERR_NO_TOKEN_SELECTED
- KMF_ERR_EXTENSION_NOT_FOUND
- KMF_ERR_DUPLICATE_KEYFILE
- KMF_ERR_OPEN_FILE
- KMF_ERR_MEMORY

2.91 kmf_get_encoded_ocsp_response

```
KMF_RETURN
kmf_get_encoded_ocsp_response(
   KMF_HANDLE_T        handle,
   char                *reqfile,
   char                *hostname,
   int                 port,
   char                *proxy,
   int                 proxy_port,
   char                *respfile,
   unsigned int        maxsecs);
```

2.91.1 Description

Applications that have an OCSP request and want to manually fetch their own OCSP response can use this function. The inputs are a filename which contains a properly encoded OCSP request, server/port, and optionally proxy/port information. Upon success, this function will dump the OCSP response received from the server into the output file indicated.

2.91.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

- KMF_ERR_CONNECT_SERVER
- KMF_ERR_OPEN_FILE
- KMF_ERR_SEND_REQUEST
- KMF_ERR_RECV_RESPONSE
- KMF_ERR_MEMORY
- KMF_ERR_RECV_TIMEOUT
- KMF_ERR_BAD_HTTP_RESPONSE
- KMF_ERR_WRITE_FILE

2.92 kmf_download_crl

```
KMF_RETURN
kmf_download_crl(
   KMF_HANDLE_T        handle,
   char                *uri,
   char                *proxy,
   int                 proxy_port,
   unsigned int        maxsecs,
   char                *crlfile,
   KMF_ENCODE_FORMAT   *pformat);
```

2.92.1 Description

This function is used to download a CRL from an external HTTP server.

2.92.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_OPEN_FILE
- KMF_ERR_MEMORY
- KMF_ERR_INTERNAL
- KMF_ERR_BAD_CRLFILE
- KMF_ERR_WRITE_FILE
- KMF_ERR_BAD_URI
- KMF_ERR_CONNECT_SERVER
- KMF_ERR_SEND_REQUEST
- KMF_ERR_RECV_RESPONSE
- KMF_ERR_RECV_TIMEOUT
- KMF_ERR_BAD_HTTP_RESPONSE

2.93 kmf_download_cert

```
KMF_RETURN
```

```
kmf_download_cert(
   KMF_HANDLE_T        handle,
   char                *uri,
   char                *proxy,
   int                 proxy_port,
   unsigned int        maxsecs,
   char                *certfile,
   KMF_ENCODE_FORMAT   *pformat);
```

2.93.1 Description

This function is used to download a certificate from an external HTTP server.

2.93.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_OPEN_FILE
- KMF_ERR_MEMORY
- KMF_ERR_INTERNAL
- KMF_ERR_BAD_CERTFILE
- KMF_ERR_WRITE_FILE
- KMF_ERR_BAD_URI
- KMF_ERR_CONNECT_SERVER
- KMF_ERR_SEND_REQUEST
- KMF_ERR_RECV_RESPONSE
- KMF_ERR_RECV_TIMEOUT
- KMF_ERR_BAD_HTTP_RESPONSE

2.94 kmf_get_ocsp_for_cert

```
KMF_RETURN
kmf_get_ocsp_for_cert(
   KMF_HANDLE_T        handle,
   KMF_DATA            *user_cert,
   KMF_DATA            *ta_cert,
   KMF_DATA            *response);
```

2.94.1 Description

This function attempts to get an OCSP response for a certificate based on either the OCSP responder extension in the user certificate or the OCSP responder URL specified in the KMF policy. The response data is returned in the KMF_DATA pointer. The memory associated with the KMF_DATA must be freed by the caller via kmf_free_data().

2.94.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_INTERNAL
- KMF_ERR_OCSP_POLICY
- KMF_ERR_BAD_URI
- KMF_ERR_PLUGIN_NOTFOUND
- KMF_ERR_FUNCTION_NOT_FOUND
- KMF_ERR_OCSP_CREATE_REQUEST
- KMF_ERR_OPEN_FILE
- KMF_ERR_ENCODING
- KMF_ERR_OCSP_BAD_ISSUER
- KMF_ERR_OCSP_BAD_CERT
- KMF_ERR_OCSP_CERTID
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.95 kmf_add_cert_eku

```
KMF_RETURN
kmf_add_cert_eku(
    KMF_X509_CERTIFICATE *certdata,
    KMF_OID              *ekuID,
    int                  critical)
```

2.95.1 Description

This function adds the Extended Key Usage OID values to a certificate record. There are some pre-defined EKU OID values defined in the libkmf header files, but this function will allow for any OID record to be used. This function checks for the presence of the EKU before adding it to avoid duplication.

2.95.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING

2.96 kmf_get_cert_auth_info_access

```
KMF_RETURN
kmf_get_cert_auth_info_access(
    const KMF_DATA             *certdata,
    KMF_X509EXT_AUTHINFOACCESS *aia)
```

2.96.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the Authority Information Access extension data in the KMF_X509EXT_AUTHINFOACCESS record if it is found.

2.96.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_EXTENSION_NOT_FOUND

2.97 kmf_get_cert_basic_constraint

```
KMF_RETURN
kmf_get_cert_basic_constraint(
    const KMF_DATA               *certdata,
    KMF_BOOL                     *critical,
    KMF_X509EXT_BASICCONSTRAINTS *constraint)
```

2.97.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the basic constraints extension data if it is found. If the basic constraints extension is found, this function also returns a boolean value to indicate whether the extension is critical or not.

2.97.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_EXTENSION_NOT_FOUND

2.98 kmf_get_cert_crl_dist_pts

```
KMF_RETURN
kmf_get_cert_crl_dist_pts(
    const KMF_DATA            *certdata,
    KMF_X509EXT_CRLDISTPOINTS *cdp)
```

2.98.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the CRL Distribution Points extension data in the KMF_X509EXT_CRLDISTPOINTS record if it is found.

2.98.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_EXTENSION_NOT_FOUND

2.99 kmf_get_cert_eku

```
KMF_RETURN
kmf_get_cert_eku(
    const KMF_DATA  *certdata,
    KMF_X509EXT_EKU *ekuptr)
```

2.99.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the extended key usage extension data if it is found.

2.99.2 Return codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_EXTENSION_NOT_FOUND

2.100 kmf_get_cert_extn

```
KMF_RETURN
kmf_get_cert_extn(
    const KMF_DATA     *certdata,
    KMF_OID            *extoid,
    KMF_X509_EXTENSION *extdata)
```

2.100.1 Description

This function takes a DER-encoded X.509 certificate data and an OID for a certificate extension as input, parses the certificate data and return the extension data if it is found.

2.100.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

- KMF_ERR_EXTENSION_NOT_FOUND

2.101 kmf_get_cert_extns

```
KMF_RETURN
kmf_get_cert_extns(
    const KMF_DATA     *certdata,
    KMF_FLAG_CERT_EXTN flag,
    KMF_X509_EXTENSION *extlist,
   int                 *nextns)
```

2.101.1 Description

This function takes a DER-encoded X.509 certificate data and an extension flag as input, searches the extensions and returns OIDs for critical, non-critical or all extensions.

The valid values for KMF_FLAG_CERT_EXTN are:

- KMF_ALL_EXTNS
- KMF_CRITICAL_EXTNS
- KMF_NONCRITICAL_EXTNS

2.101.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_EXTENSION_NOT_FOUND

2.102 kmf_get_cert_id_data

```
KMF_RETURN
kmf_get_cert_id_data(
    const KMF_DATA *certdata,
    KMF_DATA       *id)
```

2.102.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the raw data for the ID string associated with the given certificate.

2.102.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.103 kmf_get_cert_ku

```
KMF_RETURN
kmf_get_cert_ku(
    const KMF_DATA        *certdata,
    KMF_X509EXT_KEY_USAGE *keyusage)
```

2.103.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the key usage extension data if it is found.

2.103.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_EXTENSION_NOT_FOUND

2.104 kmf_get_cert_policies

```
KMF_RETURN
kmf_get_cert_policies(
    const KMF_DATA            *certdata,
    KMF_BOOL                  *critical,
    KMF_X509EXT_CERT_POLICIES *extptr)
```

2.104.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the policies extension data in the KMF_X509EXT_CERT_POLICIES record if it is found. If the policies extension is found, this function also returns a boolean value to indicate whether the extension is critical or not.

2.104.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_EXTENSION_NOT_FOUND

2.105 kmf_get_cert_validity

```
KMF_RETURN
kmf_get_cert_validity(
    const KMF_DATA *certdata,
    time_t         *not_before,
```

```
    time_t          *not_after)
```

2.105.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the NotBefore and NotAfter times (as time_t values) from the given certificate.

2.105.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING

2.106 kmf_set_cert_basic_constraint

```
KMF_RETURN
kmf_set_cert_basic_constraint(
    KMF_X509_CERTIFICATE         *cert,
    KMF_BOOL                     critical,
    KMF_X509EXT_BASICCONSTRAINTS *constraint)
```

2.106.1 Description

This function sets an X.509 v3 Basic Constraints extension to an unsigned certificate. The caller must pass in a boolean value to identify whether or not this extension is critical.

2.106.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING

2.107 kmf_set_cert_extn

```
KMF_RETURN
kmf_set_cert_extn(
    KMF_X509_CERTIFICATE *cert,
    KMF_X509_EXTENSION   *extn)
```

2.107.1 Description

This is a generic routine for adding any X509v3 extension to a certificate record. The data must be properly ASN.1 encoded in the KMF_X509_EXTENSION record (defined in header files).

2.107.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY

2.108 kmf_set_cert_issuer

```
KMF_RETURN
kmf_set_cert_issuer(
    KMF_X509_CERTIFICATE *cert,
    KMF_X509_NAME        *name)
```

2.108.1 Description

This function sets the “IssuerName” value to a certificate record. The name is of type KMF_X509_NAME. The kmf_dn_parser() function may be used to parse a character string into a valid KMF_X509_NAME record that can be used as input to this function.

2.108.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

2.109 kmf_set_cert_issuer_altname

```
KMF_RETURN
kmf_set_cert_issuer_altname(
    KMF_X509_CERTIFICATE   *cert,
    int                    critical,
    KMF_GENERALNAMECHOICES nametype,
    char                   *namedata)
```

2.109.1 Description

This function sets the “IssuerAltName” extension to a certificate record. The caller must identify whether or not this extension is critical, the type of name being used (KMF_GeneralNameChoices is defined in header files) and a pointer to a buffer containing the raw name data be to added. (See the comment in the “kmf_set_cert_subject_alname” description for the types of names that are supported.)

2.109.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CERT_FORMAT

2.110 kmf_set_cert_ku

```
KMF_RETURN
kmf_set_cert_ku(
    KMF_X509_CERTIFICATE *cert,
    int                  critical,
    uint16_t             kubits)
```

2.110.1 Description

This function sets the "Key Usage" extension to a certificate record. The caller must identify whether or not this extension is critical and pass in a bitmask of the desired key usage values.

KMF defines a list of constants that can be combined for use in the "kubits" field:

- KMF_digitalSignature
- KMF_nonRepudiation
- KMF_keyEncipherment
- KMF_dataEncipherment
- KMF_keyAgreement
- KMF_keyCertSign
- KMF_encipherOnly
- KMF_decipherOnly

Note that KMF_keyCertSign implies that the cert may also be used for signature verification as well.

2.110.1 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING

2.111 kmf_set_cert_pubkey

```
KMF_RETURN
kmf_set_cert_pubkey(
    KMF_HANDLE_T         handle,
    KMF_KEY_HANDLE       *KMFkey,
    KMF_X509_CERTIFICATE *cert)
```

2.111.1 Description

This function sets the public key to a certificate record. The KMFKey used as input is a handle to a key that was previously created or imported. This routine does not create the keypair.

2.111.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING
- KMF_ERR_NO_TOKEN_SELECTED

2.112 kmf_set_cert_serial

```
KMF_RETURN
kmf_set_cert_pubkey(
    KMF_X509_CERTIFICATE *cert,
    KMF_BIGINT           *serialNum)
```

2.112.1 Description

This function sets the serial number to a certificate record. The serial number is specified as a KMF_BIGINT structure which consists of a pointer to data and a length. This allows for values greater than 32bits.

2.112.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY

2.113 kmf_set_cert_sig_alg

```
KMF_RETURN
kmf_set_cert_sig_alg(
    KMF_X509_CERTIFICATE *cert,
    KMF_ALGORITHM_INDEX  alg)
```

2.113.1 Description

This function sets the signature algorithm that is to be used in the certificate. The Algorithm identifier is a constant that is defined in the libkmf header files. Typically, a caller choose KMF_ALGID_MD5WithRSA or KMF_ALGID_SHA1WithDSA, though others are possible.

2.113.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

2.114 kmf_set_cert_subject_altname

```
KMF_RETURN
kmf_set_cert_subject_altname(
```

```
    KMF_X509_CERTIFICATE   *cert,
    int                    critical,
    KMF_GENERALNAMECHOICES nametype,
    char                   *namedata)
```

2.114.1 Description

This function sets the “SubjectAltName” extension to a certificate record. The caller must identify whether or not this extension is critical, the type of name being used (KMF_GeneralNameChoices is defined in header files) and a pointer to a buffer containing the raw name data be to added.

KMF currently supports the following GeneralName constants for the AltName extensions:

- GENNAME_RFC822NAME – namestr is expected to be in the form of an email address (ex. Myname@sun.com)
- GENNAME_DNSNAME – namestr is expected to be in the form of a DNS hostname (ex. www.sun.com)
- GENNAME_URI – namestr is expected to be in the form of a URI (ex. Http://www.sun.com)
- GENNAME_REGISTEREDID– namestr is expected to be in the form of an object id (OID) (ex. 1.2.840.113549.1.1.4)
- GENNAME_DIRECTORYNAME – namestr is expected to be in the form of a URI (ex. Http://www.sun.com)
- GENNAME_REGISTEREDID– namestr is expected to be in the form of a directory name, like a subject name or issuer name.(ex. C=US,ST=California, L=Menlo Park, O=Sun Microsystems Inc., OU=security, CN=John Smith).

2.114.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING
- KMF_ERR_BAD_CERT_FORMAT

2.115 kmf_set_cert_validity

```
KMF_RETURN
kmf_set_cert_validity(
    KMF_X509_CERTIFICATE *cert,
    time_t               notBefore,
    uint32_t             delta)
```

2.115.1 Description

This function takes the “NotBefore” timestamp and the associated lifetime of the certificate (in seconds) as input and sets the “NotBefore” and “NotAfter” timestamps to a certificate record.

2.115.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

2.116 kmf_set_cert_version

```
KMF_RETURN
kmf_set_cert_validity(
    KMF_X509_CERTIFICATE *cert,
    uint32_t             version)
```

2.116.1 Description

This function sets the version number to a certificate record that has not yet been signed. The version value must be 0, 1 or 2 to represent the X.509 certificate version (v1 = 0, v2 = 1, v3 = 2). A value of 2 (v3) is recommended for most applications.

2.116.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

2.117 kmf_get_cert_email_str

```
KMF_RETURN
kmf_get_cert_email_str(
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.117.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the email address information (Subject AltName extension) in a character string. The caller is responsible to free the memory for the returned string after use.

2.117.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.118 kmf_get_cert_end_date_str

```
KMF_RETURN
kmf_get_cert_end_date_str(
    KMF_HANDLE_T   handle,
```

```
    const KMF_DATA *certdata,
    char           **result)
```

2.118.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the end date (NotAfter timestamp) in a character string. The caller is responsible to free the memory for the returned string after use.

2.118.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.119 kmf_get_cert_extn_str

```
KMF_RETURN
kmf_get_cert_extn_str(
    KMF_HANDLE_T       handle,
    const KMF_DATA     *certdata,
    KMF_PRINTABLE_ITEM extension,
   char               **result)
```

2.119.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns a human readable interpretation of the extended data. The caller is responsible to free the memory for the returned string after use. The following constants should be used to indicate the entension being requested:

- KMF_X509_EXT_PRIV_KEY_USAGE_PERIOD
- KMF_X509_EXT_CERT_POLICIES
- KMF_X509_EXT_SUBJ_ALTNAME
- KMF_X509_EXT_ISSUER_ALTNAME
- KMF_X509_EXT_BASIC_CONSTRAINTS
- KMF_X509_EXT_NAME_CONSTRAINTS
- KMF_X509_EXT_POLICY_CONSTRAINTS
- KMF_X509_EXT_KEY_USAGE
- KMF_X509_EXT_INHIBIT_ANY_POLICY
- KMF_X509_EXT_AUTH_KEY_ID
- KMF_X509_EXT_SUBJ_KEY_ID
- KMF_X509_EXT_POLICY_MAPPINGS
- KMF_X509_EXT_CRL_DIST_POINTS
- KMF_X509_EXT_FRESHEST_CRL

- KMF_X509_EXT_KEY_USAGE

2.119.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_ENCODING
- KMF_ERR_INTERNAL
- KMF_ERR_EXTENSION_NOT_FOUND

2.120 kmf_get_cert_id_str

```
KMF_RETURN
kmf_get_cert_id_str(
    const KMF_DATA *certdata,
    char           **idstr)
```

2.120.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns a human-readable ID string associated with a given certificate. The caller is responsible to free the memory for the returned string after use.

2.120.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.121 kmf_get_cert_issuer_str

```
KMF_RETURN
kmf_get_cert_issuer_str(
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.121.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the issuer name in a character string. The caller is responsible to free the memory for the returned string after use.

2.121.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER

- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.122 kmf_get_cert_pubkey_alg_str

```
KMF_RETURN
kmf_get_cert_pubkey_alg_str(
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.122.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the public key algorithm description in a character string. The caller is responsible to free the memory for the returned string after use.

2.122.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.123 kmf_get_cert_pubkey_str

```
KMF_RETURN
kmf_get_cert_pubkey_str(
    KMF_HANDLE_T    handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.123.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the public key data in a character string. The caller is responsible to free the memory for the returned string after use.

2.123.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.124 kmf_get_cert_serial_str

```
KMF_RETURN
kmf_get_cert_serial_str(
```

```
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.124.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the serial number in a character string. The caller is responsible to free the memory for the returned string after use.

2.124.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.125 kmf_get_cert_sig_alg_str

```
KMF_RETURN
kmf_get_cert_sig_alg_str(
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.125.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the signature algorithm description in a character string. The caller is responsible to free the memory for the returned string after use.

2.125.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.126 kmf_get_cert_start_date_str

```
KMF_RETURN
kmf_get_cert_start_date_str(
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.126.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the start date (NotBefore timestamp) in a character string. The caller is responsible

to free the memory for the returned string after use.

2.126.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.127 kmf_get_cert_subject_str

```
KMF_RETURN
kmf_get_cert_subject_str(
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.127.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the subject name in a character string. The caller is responsible to free the memory for the returned string after use.

2.127.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY
- KMF_ERR_BAD_CERT_FORMAT

2.128 kmf_get_cert_version_str

```
KMF_RETURN
kmf_get_cert_version_str(
    KMF_HANDLE_T   handle,
    const KMF_DATA *certdata,
    char           **result)
```

2.128.1 Description

This function takes a DER-encoded X.509 certificate data as input and returns the version number in a character string. The caller is responsible to free the memory for the returned string after use.

2.128.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_MEMORY

- KMF_ERR_BAD_CERT_FORMAT

2.129 kmf_build_pk12

```
KMF_RETURN
kmf_build_pk12(
   KMF_HANDLE_T      handle,
   int               numcerts,
   KMF_X509_DER_CERT *certlist,
   int               numkeys,
   KMF_KEY_HANDLE    *keylist,
   KMF_CREDENTIAL    *p12cred,
   char              *filename)
```

2.129.1 Description

This function takes a list of raw certificate data and the raw key data of their associated private key as input, then build a PKCS#12 PDU file from them.

2.129.2 Return Codes

- KMF_OK
- KMF_ERR_BAD_PARAMETER
- KMF_ERR_OPEN_FILE
- KMF_ERR_ENCODING
- KMF_ERR_MEMORY
- KMF_ERR_CERT_NOT_FOUND
- KMF_ERR_KEY_NOT_FOUND

2.130 kmf_check_cert_date

```
KMF_RETURN
kmf_check_cert_date(
    KMF_HANDLE_T   handle,
    const KMF_DATA *cert)
```

2.130.1 Description

This function checks the validity period of a DER encoded certificate. If there is a validity period adjustment associated with the current policy that is tied tot he KMF handle used in this call, then the adjustment factor is used when checking the validity period. The adjustment period is specified as the “validity_adjusttime” parameter when creating or modifying policies with the kmfcfg utility and is applied to both the NotBefore and NotAfter dates in the certificate.

2.130.2 Return Codes

- KMF_OK

- KMF_ERR_BAD_PARAMETER
- KMF_ERR_BAD_CERT_FORMAT
- KMF_ERR_VALIDITY_PERIOD

3