THALES

CipherTrust Application Data Protection CAPI

API GUIDE

Software Version: 8.13

Preface

All information herein is either public information or is the property of and owned solely by Thales and/or its subsidiaries or affiliates who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Thales's information.

This document can be used for informational, non-commercial, and personal use only provided that:

- The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any publicly accessible network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Thales makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Thales reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Thales hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Thales be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Thales does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Thales be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Thales products. Thales disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

Support Contacts

If you encounter a problem while installing, registering, or operating this product, please refer to the documentation before contacting support. If you cannot resolve the issue, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable database of support resources, including:

- Software and firmware downloads
- Latest product documentation
- Latest release notes listing known problems and workarounds
- A knowledge base
- FAQs
- Technical notes, and more

You can also use the portal to create and manage support cases.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.dis@thalesgroup.com.

Table of Contents

1 Overview

The CADP CAPI API is divided into the following categories:

- **Library Management API:** Can be used for library management requirements such as initialization and session handling etc.
- **NAE Key Management API:** Deals with Key Management using the NAE XML protocol.
- **KMIP Key Management API:** Exposes functionality using the KMIP Protocol.
- **Cryptographic API**: Can be used to perform local cryptographic operations after fetching keys from CipherTrust Manager. However, CADP CAPI client cannot offload cryptographic operations to CipherTrust Manager.

When creating ICAPI Objects, indicated by the prefix `I_O_` or `I_KO_`, you do not need to allocate memory. Memory is internally allocated when these objects are passed to a function.

For example, you can declare a new key info object as follows:

```
I_O_KeyInfo ki;
```

When the `I_CreateKeyInfo` function is called, the system allocates and then uses the memory needed for `ki`.

Identifier Naming Convention

The following identifier naming convention is used:

- `I_C_` denotes a function.
- `I_O_` denotes an opaque object.
- `I_T_` denotes a type or enumeration.
- `I_E_` denotes an enumeration value for error code.
- `I_KC_` denotes a KMIP Key Management function.
- `I_KO_` denotes a KMIP Key Management Opaque Object.
- `I_KT_` denotes a KMIP Key Management type or enumeration.
- `I_KS_` denotes a KMIP Structure type.

2 Symbol Reference

2.1 Functions

The following table lists functions in this documentation.

Functions

	Name	Description
	Decrypt	Decrypt inData : input data , this is value filled in outdata parameter of Encrypt() API. length : Length of inData outputData : Address of unallocated buffer, please set *outputData = NULL before sending it. sess : An CAPI open session. freeIndata : set it 0 id user will free inData themselves, otherwise set 1 Decrypt() will free it. Return: length of outputData and 0 in case of error.
	Encrypt	Encrypt keyName: name of key inData: input data to encrypt datalen: length of in data outdata: Address of unallocated buffer, please set *outdata = NULL before sending it. sess : An CAPI open session. Return: length of outdata and 0 in case of error.
	I_C_AddGroupToObject	This function adds a user group to a groupList object.
	I_C_AddToAttributeList	This function adds an attribute to an existing custom attribute list.
	I_C_AddToAttributeListWithType	This function adds an attribute to an existing custom attribute list.
	I_C_CloneKey	This function generates a new key based on the key bytes of another key.
	I_C_CreateCertificateRequest	This function creates a CSR on KS.
	I_C_CreateCertificateSignRequest	This function used to Sign the CSR Request on KS. *... more
	I_C_CreateCipherSpec	This function creates a CipherSpec object. A CipherSpec defines an algorithm and a key. It may be reused in multiple cryptographic operations and may be used in more than one operation at a time.

	I_C_CreateKey	This function creates a key. To create a versioned key, append a # to the end of the keyName parameter. This feature is for versions of the NAE server that support versioned keys.
	I_C_CreateKeyInfo	This function creates a KeyInfo object. **Note:** Creating a 2048-bit RSA key takes more than 1 second. If your Connection_Timeout property is set to a value less than 2 or 3 seconds, 2048-bit key creation could return an error *even though the key is successfully created*.
	I_C_CreateKeyInfo_KeyDetails	This function creates a KeyInfo object for algorithms like EC which have additional attributes such as curveID.
	I_C_Crypt	This function encrypts data in a single chunk. Use I_C_Crypt to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt blocks while waiting for the results.
	I_C_CryptAllVersions	This function encrypts data with all active versions of a key. Use I_C_CryptAllVersions to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_CryptAllVersions blocks while waiting for the results.
	I_C_CryptBulk	This function encrypts an array of data elements. Use the Bulk interface to operate on a large array of data elements using the same key. Bulk is optimized for high throughput where latency is not a priority. If the ivFlag is I_T_IV_PerElement, then there should be the same number of IVs as the number of inData elements. If the ivFlag is I_T_IV_Single, then there should be one IV. **Note:** While using the I_C_CryptBulk API, a Connection_Timeout value of 60000 milliseconds is recommended. The default value, 30000 milliseconds, may not be appropriate when the number of operations being performed is high.... more

	I_C_CryptBulk_Enhanced	This I_C_CryptBulk_Enhanced function encrypts array of data elements using the same key. return : ERROR if all data elements are not operated successfully If the ivFlag is I_T_IV_PerElement, then there should be the same number of IVs as the number of inData elements. If the ivFlag is I_T_IV_Single, then there should be one IV. If the USpecFlag is I_T_Uspec_PerElement, then there should be the same number of userspecs as the number of inData elements. If the USpecFlag is I_T_Uspec_Single, then there should be one userspec. **Note:** The user must use same set of valid userspec input values for each element,... more
	I_C_CryptInit	This function determines the cryptographic operation, iv, and cipher to be used for the subsequent calls to I_C_CryptUpdate. Use the I_C_CryptInit/I_C_CryptUpdate/I_C_CryptFinal interface (multiple updates are OK) when you want results back from part of your cryptographic operation before you have all the data ready, or if your data is larger than I_C_Crypt will allow. I_C_CryptUpdate and I_C_CryptFinal block while waiting for the results.
	I_C_CryptUpdate	This function sends the input data and receives the output data. You can use this function to get the results of your cryptographic operation before entering all of your data. Use the I_C_CryptInit/I_C_CryptUpdate/I_C_CryptFinal interface (multiple updates are OK) when you want results back from part of your cryptographic operation before you have all the data ready, or if your data is larger than I_C_Crypt will allow. I_C_CryptUpdate and I_C_CryptFinal block while waiting for the results.
	I_C_Crypt_Enhanced	This function encrypts data in a single chunk. Use I_C_Crypt_Enhanced to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt_Enhanced blocks while waiting for the results.
	I_C_Crypt_Enhanced_FpeFormat	This function encrypts data in a single chunk. Use I_C_Crypt_Enhanced to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt_Enhanced blocks while waiting for the results.

	I_C_Crypt_Fast	This function encrypts data in a single chunk. Use I_C_Crypt_Fast to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt_Fast blocks while waiting for the results. This will work only when symmetric cache is on. This API gives better performance than I_C_Crypt.
	I_C_DestroyCertificate	This function destroys a certificate.
	I_C_DestroyKey	This function destroys a key on the cluster of servers. **Important!** Once you destroy a key, any values encrypted by that key are forever lost.
	I_C_ExportAESWrappedKey	This function exports key bytes of a symmetric key in a wrapped form. The user must be the owner of the key or must have the permission to export the key. The wrap format specifies the algorithm used to wrap the plain key.
	I_C_ExportCAChain	This function exports a CA chain from the DataSecure to a specified format.
	I_C_ExportCertificate	This function exports a certificate or a certificate/key combination from the DataSecure to a specified format. The certificate or certificate/key combination is in one of the PEM or PKCS formats defined in I_T_ExportFormat.
	I_C_ExportKey	This function Export Key according to key type and format specified by user
	I_C_ExportPublicKey	This function exports the public portion of an RSA key pair.
	I_C_ExportSymmetricKey	This function exports key bytes of a symmetric key. The user must be the owner of the key or must have permission to export the key.

	I_C_ExportWrappedKey	This function exports key bytes of a symmetric key in a wrapped form. The user must be the owner of the key or must have the permission to export the key. The wrap format specifies the algorithm used to wrap the plain key. I_T_ExportKeyWrapFormat_RAW_PKCS1v15 ,I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA256 , I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA384 , I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA512 are the wrap formats supported. When the format given is I_T_ExportKeyWrapFormat_RAW_PKCS1v15 , the key bytes are encrypted using RSA public key using PKCS1v1.5 format I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA256, key bytes are encrypted using RSA public key using PKSC1v2.1/RSAOAEP-SHA256 format. I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA384 , key bytes are encrypted using RSA public key using PKCS1v2.1/RSAOAEP-SHA384 format. I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA512 ,... more
	I_C_FindInAttributeList	This function finds an attribute value in an attribute list.
	I_C_FindInAttributeListwithType	This function finds an attribute value and type in an attribute list.
	I_C_FindInstanceInAttributeList	This function retrieves the value of a specific instance of an attribute with the given attributeName. This function was designed to be used with an attributeList that may contain multiple instances of attributes with the same attributeName. Typically, it is used to retrieve the values of all instances of an attributeName.
	I_C_FindKey	This function returns a list of keys having certain attiribute This function allocates memory for pKeyNamesList, therefore the user must call later I_C_DeleteStringList(pStringList); to prevent a memory leak
	I_C_GetCiphertextHeaderLength	This function returns the length of the ciphertext's header. You can use this function to get data about both versioned and non-versioned keys.

	I_C_GetKeyAttributes	This function returns the attributes of a key. Key attributes include keysize, permissions, algorithm, deletable, exportable, and any user-defined elements. **Note:** The user must be the owner of the key, or must have access granted to the key.
	I_C_GetKeyNames	This function get Key Names according to custom attributes listed or fingerprint
	I_C_GetUserAttributes	This function returns the attributes of a user.
	I_C_GetUserSpec	This function get values from a UserSpec object. A UserSpec defines additional User inputs. It may be reused in multiple cryptographic operations and may be used in more than one operation at a time.
	I_C_ImportCertificate	This function imports a certificate or certificate/key combination into the DataSecure. The DataSecure will determine the certificate's format based on the data itself. If the certificate is in PKCS#12 format, you must provide the password.
	I_C_ImportKey	This function imports a key to the server. **Note:** The versioned keys cannot be imported.
	I_C_Initialize	This function initializes the library.
	I_C_LogEvent	This function logs a message on the server.
	I_C_ModifyGroupPermissions	This function modifies the permissions of a group for the specified key.
	I_C_OpenSession	This function opens a new session. If KMIP is configured (by configuring the KMIP_IP parameter in the properties file), using I_C_OpenSession with the I_T_Auth_Password parameter returns failure. This happens because the Credential Base Object is not supported. You can use the I_T_AuthNoPassword parameter in this case.
	I_C_OpenSessionPersistentCacheCallback	This function opens a new session supplying a persistent cache callback function.
	I_C_OpenSessionPersistentCacheCallback_filepath	This function opens a new session supplying a persistent cache callback function.
	I_C_OpenSessionPersistentCacheParameters	This function opens a new session with a persistent cache certificate used for retrieving passphrase.
	I_C_OpenSessionPersistentCachePassphrase	This function opens a new session with a persistent cache passphrase.
	I_C_OpenSessionPersistentCachePassphrase_filepath	This function opens a new session with a persistent cache passphrase.

	I_C_OpenSession_filepath	This function opens a new session. If KMIP is configured (by configuring the KMIP_IP parameter in the properties file), using I_C_OpenSession_filepath with the I_T_Auth_Password parameter returns failure. This happens because the Credential Base Object is not supported. You can use the I_T_AuthNoPassword parameter in this case.
	I_C_RemoveFromAttributeList	This function removes an attribute from an I_O_AttributeList object.
	I_C_SetKeyAttributes	This function sets custom attributes for the key on the server. Only the owner of the key may modify the attributes.
	I_C_SetKeyParameter	This function can modify the state of a key version or create a new version. **Note:** This function only supports versioned keys. You cannot create a new version of a nonversioned key.
	I_C_SetUserSpec	This function creates and Set values in a UserSpec object. A UserSpec defines additional User inputs. It may be reused in multiple cryptographic operations and may be used in more than one operation at a time.
	I_KC_AddToAttributeList	This function adds attributes to the Attribute List. The added attributes will be sent in KMIP Request.
	I_KC_Create	This function creates the Symmetric Key Object with the KMIP server.
	I_KC_CreateAttributeList	This function creates a new Attribute List for use with KMIP Operations.
	I_KC_Crypto	This function will encrypt the data given by user
	I_KC_FreeCryptoObject	This function retrieves the attributes of the Managed Object from the KMIP server.
	I_KC_Get	This function retrieves a Managed Object from the Key Management Server.
	I_KC_GetAttributes	This is function I_KC_GetAttributes.
	I_KC_GetResultReasonString	This function returns the result reason.
	I_KC_GetResultStatusString	This function returns the result status.
	I_KC_GetWrappedKey	This function get wrapped key bytes for a Managed Object from the Key Management Server.
	I_KC_Locate	This function locates Managed Objects as per the specified search criteria.

	I_KC_Query	This function queries the server of its capabilites. I_C_FreeQueryResponse should be used to free up the memory allocated by the library, to hold the response. The queryFunctions parameter can be used to specify the requested capabilities. I_KT_QueryFunction_ApplicationNameSpaces response is not supported.
	I_KC_ReKey	This function Recreate the Symmetric Key Object with the KMIP server.
	I_KC_ReKeyPair	This function Recreate the Asymmetric Key Object with the KMIP server.
	I_KC_Register	This function registers the Managed Object with the KMIP server.
	I_KC_RegisterAsymmetricKey	This function registers the Managed Object of type Asymmestric Key with the KMIP server.
	I_KC_RetrieveFromAttributeList	This function retrieves attribute(s) from the Attribute List.

2.1.1 Decrypt Function

C/C++

```
int Decrypt(unsigned char* inData, int length, unsigned char** outputData,
I_O_Session sess, int freeIndata);
```

File

File: CryptoDataUtility.h

Description

Decrypt inData : input data , this is value filled in outdata parameter of Encrypt() API. length : Length of inData outputData : Address of unallocated buffer, please set *outputData = NULL before sending it. sess : An CAPI open session. freeIndata : set it 0 id user will free inData themselves, otherwise set 1 Decrypt() will free it.

Return: length of outputData and 0 in case of error.

2.1.2 Encrypt Function

C/C++

```
int Encrypt(char* keyName, unsigned char* inData, unsigned int datalen, unsigned
char** outdata, I_O_Session sess);
```

File

File: CryptoDataUtility.h

Description

Encrypt keyName: name of key inData: input data to encrypt datalen: length of in data outdata: Address of

unallocated buffer, please set *outdata = NULL before sending it. sess : An CAPI open session.

Return: length of outdata and 0 in case of error.

2.1.3 I_C_AddGroupToObject Function

C/C++

```
I_T_RETURN FUNCEXP I_C_AddGroupToObject(I_O_GroupList groupList, const I_T_CHAR *
groupName, I_T_UINT permissionMask);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_GroupList groupList	The GroupList to which the group will be added.
const I_T_CHAR * groupName	The name of the group.
I_T_UINT permissionMask	The permissions granted to the group. For example, I_T_Permission_Encrypt, I_T_Permission_Decrypt. These are listed in "I_T_PermissionMaskEnum".

Description

This function adds a user group to a groupList object.

2.1.4 I_C_AddToAttributeList Function

C/C++

```
I_T_RETURN FUNCEXP I_C_AddToAttributeList(I_O_AttributeList customAttributeList,
const I_T_CHAR * attributeName, const I_T_CHAR * attributeValue);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList customAttributeList	An attribute list object to which an attribute should be added. Note that only a custom attribute list may be passed.
const I_T_CHAR * attributeName	The name of the attribute. The name must be null terminated strings of at most 64 characters (excluding null termination) and may only contain the following characters:

const I_T_CHAR * attributeValue	The value of the attribute. A new attribute will be added to the attribute list if an attribute with the attributeName does not already exist. Otherwise, the existing value will be overwritten with attributeValue. The value must not contain more than 1024 characters, and must be null-terminated strings of 7-bit US ASCII characters.

Description

This function adds an attribute to an existing custom attribute list.

2.1.5 I_C_AddToAttributeListWithType Function

C/C++

```
I_T_RETURN FUNCEXP I_C_AddToAttributeListWithType(I_O_AttributeList
customAttributeList, const I_T_CHAR * attributeName, const I_T_CHAR *
attributeValue, const I_T_INT attributeDataType);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList customAttributeList	An attribute list object to which an attribute should be added. Note that only a custom attribute list may be passed.
const I_T_CHAR * attributeName	The name of the attribute. The name must be null terminated strings of at most 64 characters (excluding null termination) and may only contain the following characters:
const I_T_CHAR * attributeValue	The value of the attribute. A new attribute will be added to the attribute list if an attribute with the attributeName does not already exist. Otherwise, the existing value will be overwritten with attributeValue. The value must not contain more than 1024 characters, and must be null-terminated strings of 7-bit US ASCII characters.
const I_T_INT attributeDataType	The datatype of the attribute. The default value of the attribute is -1, that means if user doesn't mention any datatype, then by defalut datatype of the attribute will be string.

Description

This function adds an attribute to an existing custom attribute list.

2.1.6 I_C_CloneKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CloneKey(I_O_Session handle, const I_T_CHAR * keyName, const
```

```
I_T_CHAR * newKeyName);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
const I_T_CHAR * keyName	The name of the key to be cloned.
const I_T_CHAR * newKeyName	The name of the new clone.
session	The current session.

Description

This function generates a new key based on the key bytes of another key.

2.1.7 I_C_CreateCertificateRequest Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateCertificateRequest(I_O_Session sessionHandle, const
I_T_CHAR * keyName, const I_T_CHAR * commonName, const I_T_CertificateDetails *
certDetail, I_T_CHAR ** data, I_T_UINT * dataSize);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session sessionHandle	The current session.
const I_T_CHAR * keyName	The name for the key. key name for an existing asymmetric key
const I_T_CHAR * commonName	common name for the certificate is mandatory parameter.
const I_T_CertificateDetails * certDetail	Certificate parameters . this parameter contain all the supported Certificate papameters. Country name is mandatory parameters.
I_T_CHAR ** data	A pointer to Buffer to hold the returned Certificate content.
I_T_UINT * dataSize	A pointer to hold the length of the Certificate.

Description

This function creates a CSR on KS.

2.1.8 I_C_CreateCertificateSignRequest Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateCertificateSignRequest(I_O_Session sessionHandle, const
I_T_CHAR * caName, I_T_UINT certificateUsage, I_T_UINT certificateExpiry, const
```

```
I_T_CHAR * csrData, I_T_UINT csrDataLen, I_T_CHAR ** signdata, I_T_UINT *
signdataSize);
```

File

File: cadp_capi_naekeymgmt.h

Description

This function used to Sign the CSR Request on KS. *

- param sessionHandle The current session.
- param caName The name for the CA.

*

- param certificateUsage .

*

- Indicates whether the certificate is used for a Client, the Server, or an Intermediate CA.

*

- param certificateExpiry.
- Contains the certificate expiry time in days.

*

- param csrData Contains the certificate signing request data.

*

- param csrDataLen Contains the Length of certificate signing request data.

* *param signdata Contains the certificate signing response data. * *param signdataSize Contains the length of signing response data. * *

2.1.9 I_C_CreateCipherSpec Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateCipherSpec(const I_T_CHAR * longAlgorithmName, const
I_T_CHAR * keyName, I_O_CipherSpec * cipher);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
const I_T_CHAR * longAlgorithmName	The full algorithm name, for example, "AES/CBC/PKCS5Padding".
const I_T_CHAR * keyName	The ASCII key name.
I_O_CipherSpec * cipher	The cipher that will be created. The pointer to an I_O_CipherSpec to hold the returned object.

Description

This function creates a CipherSpec object.

A CipherSpec defines an algorithm and a key. It may be reused in multiple cryptographic operations and may be used in more than one operation at a time.

2.1.10 I_C_CreateKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateKey(I_O_Session session, const I_T_CHAR * keyName,
I_O_KeyInfo keyInfo, I_O_GroupList groupList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * keyName	The name for the new key.
I_O_KeyInfo keyInfo	The KeyInfo object (algorithm name, key size, etc) for the key. See I_C_CreateKeyInfo.
I_O_GroupList groupList	The GroupList object that will have access to the key. See I_C_CreateGroupListObject.

Description

This function creates a key.

To create a versioned key, append a # to the end of the keyName parameter. This feature is for versions of the NAE server that support versioned keys.

2.1.11 I_C_CreateKeyInfo Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateKeyInfo(const I_T_CHAR * shortAlgorithmName, I_T_UINT
keySizeInBits, I_T_BOOL exportable, I_T_BOOL deletable, I_O_KeyInfo * keyInfo);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
const I_T_CHAR * shortAlgorithmName	A pointer to the algorithm name, such "AES" or "DES".
I_T_UINT keySizeInBits	The key size in bits. Use a key size of 168 for triple DES.

I_T_BOOL exportable	If true, this parameter allows the key to be exported from a non-FIPS device.
I_T_BOOL deletable	If true, this parameter allows the key to be deleted.
I_O_KeyInfo * keyInfo	A pointer to an I_O_KeyInfo to hold the returned object.

Description

This function creates a KeyInfo object.

Note: Creating a 2048-bit RSA key takes more than 1 second. If your Connection_Timeout property is set to a value less than 2 or 3 seconds, 2048-bit key creation could return an error *even though the key is successfully created*.

2.1.12 I_C_CreateKeyInfo_KeyDetails Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateKeyInfo_KeyDetails(const I_T_CHAR * shortAlgorithmName,
I_T_UINT keySizeInBits, I_T_BOOL exportable, I_T_BOOL deletable, I_O_KeyInfo *
keyInfo, I_T_KeyDetails * keyDetails);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
const I_T_CHAR * shortAlgorithmName	A pointer to the algorithm name, e.g."EC"
I_T_UINT keySizeInBits	The key size in bits. Use a key size of 168 for triple DES.
I_T_BOOL exportable	If true, this parameter allows the key to be exported from a non-FIPS device.
I_T_BOOL deletable	If true, this parameter allows the key to be deleted.
I_O_KeyInfo * keyInfo	A pointer to an I_O_KeyInfo to hold the returned object.
curveID	A pointer to the curve ID.

Description

This function creates a KeyInfo object for algorithms like EC which have additional attributes such as curveID.

2.1.13 I_C_Crypt Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Crypt(I_O_Session session, I_O_CipherSpec cipher,
I_T_Operation operation, const I_T_BYTE * iv, I_T_UINT ivLen, const I_T_BYTE *
inData, I_T_UINT inDataLen, I_T_BYTE * outData, I_T_UINT * outDataLen);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_T_Operation operation	The cryptographic operation to perform.
const I_T_BYTE * iv	The initialization vector used for CBC mode block ciphers.
I_T_UINT ivLen	The length of the IV. If you want to use default IVs in the CBC mode, set ivLen=0 and iv as null.
const I_T_BYTE * inData	The data to encrypt or decrypt.
I_T_UINT inDataLen	The length of the input data.
I_T_BYTE * outData	A buffer to hold the output data.
I_T_UINT * outDataLen	The length of the output data. [in] The length of outData. [out] The number of bytes returned.

Description

This function encrypts data in a single chunk.

Use I_C_Crypt to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt blocks while waiting for the results.

2.1.14 I_C_CryptAllVersions Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CryptAllVersions(I_O_Session session, I_O_CipherSpec cipher,
I_T_Operation operation, I_T_UINT* numKeyVersions, const I_T_BYTE* iv, I_T_UINT
ivLen, const I_T_BYTE* inData, I_T_UINT inDataLen, I_T_BYTE** outData, I_T_UINT*
outDataLen);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_T_Operation operation	The cryptographic operation to perform (encrypt only).
I_T_UINT* numKeyVersions	The number of elements in output data buffer. To determine the value of numKeyVersions (i.e., the number of active versioned keys) if unknown to the programmer, call I_C_CryptAllVersions with numKeyVersions == 0, iv, inData, outData, and outDataLen == NULL, and inDataLen == 0. On return, *numKeyVersions will have the number of active keys.
const I_T_BYTE* iv	The initialization vector; used for CBC mode block ciphers.

I_T_UINT ivLen	The length of the IV.
const I_T_BYTE* inData	The data to encrypt or decrypt.
I_T_UINT inDataLen	The length of the input data.
I_T_BYTE** outData	A buffer to hold the output data.
I_T_UINT* outDataLen	The length of the output data. [in] The length of outData. [out] The number of bytes returned.

Description

This function encrypts data with all active versions of a key.

Use I_C_CryptAllVersions to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_CryptAllVersions blocks while waiting for the results.

2.1.15 I_C_CryptBulk Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CryptBulk(I_O_Session session, I_O_CipherSpec cipher,
I_T_Operation operation, I_T_UINT numOps, I_T_IVType ivFlag, const I_T_BYTE ** ivs,
I_T_UINT ivLen, const I_T_BYTE ** inData, I_T_UINT * inDataLen, I_T_BYTE ** outData,
I_T_UINT * outDataLen);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_T_Operation operation	The operation that will be performed.
I_T_UINT numOps	The number of operations that will be performed.
I_T_IVType ivFlag	This is the method of applying the IV, either per element (I_T_IV_PerElement) or single IV (I_T_IV_Single). When applying IVs per element, there must be an IV for each inData element. When the flag is set to I_T_IV_Single, there should be one IV.
const I_T_BYTE ** ivs	A pointer to the initialization vector; used for CBC mode block ciphers.
I_T_UINT ivLen	The length of the IV.
const I_T_BYTE ** inData	A data array containing the data to be encrypted or decrypted.
I_T_UINT * inDataLen	A data length array containing the length of the input data.
I_T_BYTE ** outData	A data array containing the resulting data.
I_T_UINT * outDataLen	A data length array containing the length of the output data.

Description

This function encrypts an array of data elements.

Use the Bulk interface to operate on a large array of data elements using the same key. Bulk is optimized for high throughput where latency is not a priority. If the ivFlag is I_T_IV_PerElement, then there should be the same number of IVs as the number of inData elements. If the ivFlag is I_T_IV_Single, then there should be one IV.

Note: While using the I_C_CryptBulk API, a Connection_Timeout value of 60000 milliseconds is recommended. The default value, 30000 milliseconds, may not be appropriate when the number of operations being performed is high.

2.1.16 I_C_CryptBulk_Enhanced Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CryptBulk_Enhanced(I_O_Session handle, I_O_CipherSpec cipher,
I_T_Operation operation, I_T_UINT numOps, I_T_IVType IVFlag, const I_T_BYTE ** iv,
I_T_UINT ivlen, const I_T_BYTE ** InData, I_T_UINT * InDataLen, I_T_BYTE ** OutData,
I_T_UINT * OutDataLen, I_T_UspecType USpecFlag, I_O_UserSpec* userspec);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session handle	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_T_Operation operation	The operation that will be performed.
I_T_UINT numOps	The number of operations that will be performed.
I_T_IVType IVFlag	This is the method of applying the IV, either per element (I_T_IV_PerElement) or single IV (I_T_IV_Single). When applying IVs per element, there must be an IV for each inData element. When the flag is set to I_T_IV_Single, there should be one IV.
const I_T_BYTE ** iv	A pointer to the initialization vector; used for CBC mode block ciphers.
I_T_UspecType USpecFlag	This is the method of applying the userspec, either per element (I_T_Uspec_PerElement) or single userspec (I_T_Uspec_Single). When applying userspecs per element, there must be an userspec for each inData element. When the flag is set to I_T_Uspec_Single, there should be one userspec.
I_O_UserSpec* userspec	Additional user inputs already set in userspec(s), It must be same set of user inputs values
ivLen	The length of the IV.
inData	A data array containing the data to be encrypted or decrypted.

inDataLen	A data length array containing the length of the input data.
outData	A data array containing the resulting data.
outDataLen	A data length array containing the length of the output data.

Description

This I_C_CryptBulk_Enhanced function encrypts array of data elements using the same key. return : ERROR if all data elements are not operated successfully If the ivFlag is I_T_IV_PerElement, then there should be the same number of IVs as the number of inData elements. If the ivFlag is I_T_IV_Single, then there should be one IV. If the USpecFlag is I_T_Uspec_PerElement, then there should be the same number of userspecs as the number of inData elements. If the USpecFlag is I_T_Uspec_Single, then there should be one userspec.

Note: The user must use same set of valid userspec input values for each element, in case USpecFlag is I_T_Uspec_PerElement.For example ,if using tweakAlgo is "SHA1" for one element, than tweakAlgo must not be NULL or any other value for successive userspec elements.

2.1.17 I_C_CryptInit Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CryptInit(I_O_Session session, I_O_CipherSpec cipher,
I_T_Operation operation, const I_T_BYTE * iv, I_T_UINT ivLen, I_O_CipherState *
state);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_T_Operation operation	The operation that will be performed.
const I_T_BYTE * iv	The initialization vector; used for CBC mode block ciphers.
I_T_UINT ivLen	The length of the iv.
I_O_CipherState * state	An object containing information about the cryptographic operation, such as bytes sent.

Description

This function determines the cryptographic operation, iv, and cipher to be used for the subsequent calls to I_C_CryptUpdate.

Use the I_C_CryptInit/I_C_CryptUpdate/I_C_CryptFinal interface (multiple updates are OK) when you want results back from part of your cryptographic operation before you have all the data ready, or if your data is larger than I_C_Crypt will allow. I_C_CryptUpdate and I_C_CryptFinal block while waiting for the results.

2.1.18 I_C_CryptUpdate Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CryptUpdate(I_O_Session session, I_O_CipherState state, const
I_T_BYTE * inData, I_T_UINT inDataLen, I_T_BYTE * outData, I_T_UINT * outDataLen);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherState state	An object with information about the cryptographic operation, such as bytes sent.
const I_T_BYTE * inData	The data to be encrypted or decrypted.
I_T_UINT inDataLen	The length of the input data.
I_T_BYTE * outData	The resulting data.
I_T_UINT * outDataLen	The length of the output data.

Description

This function sends the input data and receives the output data. You can use this function to get the results of your cryptographic operation before entering all of your data.

Use the I_C_CryptInit/I_C_CryptUpdate/I_C_CryptFinal interface (multiple updates are OK) when you want results back from part of your cryptographic operation before you have all the data ready, or if your data is larger than I_C_Crypt will allow. I_C_CryptUpdate and I_C_CryptFinal block while waiting for the results.

2.1.19 I_C_Crypt_Enhanced Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Crypt_Enhanced(I_O_Session session, I_O_CipherSpec cipher,
I_T_Operation operation, const I_T_BYTE * iv, I_T_UINT ivLen, const I_T_BYTE *
inData, I_T_UINT inDataLen, I_T_BYTE * outData, I_T_UINT * outDataLen, I_O_UserSpec
userspec);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_T_Operation operation	The cryptographic operation to perform.

const I_T_BYTE * iv	The initialization vector used for CBC mode block ciphers.
I_T_UINT ivLen	The length of the IV. If you want to use default IVs in the CBC mode, set ivLen=0 and iv as null.
const I_T_BYTE * inData	The data to encrypt or decrypt.
I_T_UINT inDataLen	The length of the input data.
I_T_BYTE * outData	A buffer to hold the output data.
I_T_UINT * outDataLen	The length of the output data. [in] The length of outData. [out] The number of bytes returned.
I_O_UserSpec userspec	The userspec that will be created. The pointer to an I_O_UserSpec to hold the returned object.

Description

This function encrypts data in a single chunk.

Use I_C_Crypt_Enhanced to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt_Enhanced blocks while waiting for the results.

2.1.20 I_C_Crypt_Enhanced_FpeFormat Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Crypt_Enhanced_FpeFormat(I_O_Session session, I_O_CipherSpec
cipher, I_T_Operation operation, const I_T_BYTE * iv, I_T_UINT ivLen, const I_T_BYTE
* inData, I_T_UINT inDataLen, I_T_BYTE * outData, I_T_UINT * outDataLen,
I_O_UserSpec userspec, I_T_FpeFormat_Type fpeFormat);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_T_Operation operation	The cryptographic operation to perform.
const I_T_BYTE * iv	The initialization vector used for CBC mode block ciphers.
I_T_UINT ivLen	The length of the IV. If you want to use default IVs in the CBC mode, set ivLen=0 and iv as null.
const I_T_BYTE * inData	The data to encrypt or decrypt.
I_T_UINT inDataLen	The length of the input data.
I_T_BYTE * outData	A buffer to hold the output data.
I_T_UINT * outDataLen	The length of the output data. [in] The length of outData. [out] The number of bytes returned.
I_O_UserSpec userspec	The userspec that will be created. The pointer to an I_O_UserSpec to hold the returned object.
I_T_FpeFormat_Type fpeFormat	The format type for FPE

Description

This function encrypts data in a single chunk.

Use I_C_Crypt_Enhanced to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt_Enhanced blocks while waiting for the results.

2.1.21 I_C_Crypt_Fast Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Crypt_Fast(I_O_Session session, I_O_CipherSpec cipher,
I_O_CipherState * state, I_T_Operation operation, const I_T_BYTE * iv, I_T_UINT
ivLen, const I_T_BYTE * inData, I_T_UINT inDataLen, I_T_BYTE * outData, I_T_UINT *
outDataLen);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be used for the operation.
I_O_CipherState * state	The cipher state structure which holds connection object which contains information about the cryptographic operation such as bytes sent.
I_T_Operation operation	The cryptographic operation to perform.
const I_T_BYTE * iv	The initialization vector used for CBC mode block ciphers.
I_T_UINT ivLen	The length of the IV. If you want to use default IVs in the CBC mode, set ivLen=0 and iv as null.
const I_T_BYTE * inData	The data to encrypt or decrypt.
I_T_UINT inDataLen	The length of the input data.
I_T_BYTE * outData	A buffer to hold the output data.
I_T_UINT * outDataLen	The length of the output data. [in] The length of outData. [out] The number of bytes returned.

Description

This function encrypts data in a single chunk.

Use I_C_Crypt_Fast to encrypt complete chunks of data less than 3K bytes when you want the results immediately. I_C_Crypt_Fast blocks while waiting for the results. This will work only when symmetric cache is on. This API gives better performance than I_C_Crypt.

2.1.22 I_C_DestroyCertificate Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DestroyCertificate(I_O_Session sessionHandle, const I_T_CHAR
* certificateName);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session sessionHandle	The current session.
const I_T_CHAR * certificateName	The name of the certificate to be destroyed.

Description

This function destroys a certificate.

2.1.23 I_C_DestroyKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DestroyKey(I_O_Session session, const I_T_CHAR * keyName);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * keyName	The name of the key that will be irretrievably destroyed.

Description

This function destroys a key on the cluster of servers.

Important! Once you destroy a key, any values encrypted by that key are forever lost.

2.1.24 I_C_ExportAESWrappedKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ExportAESWrappedKey(I_O_Session handle, const I_T_CHAR *
keyName, I_T_UINT KeyVersion, unsigned int * numVersion, const I_T_BYTE* wrapKey,
const I_T_AESKeyWrapFormat aesWrapFormat, I_T_BYTE *** ppWrappedKeyBytes, unsigned
int ** pWrappedKeyBytesLen);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	The handle to the session.
const I_T_CHAR * keyName	The name of the symmetric key that should be exported. The key must be exportable.
I_T_UINT KeyVersion	This parameter have 3 values:
unsigned int * numVersion	This is out parameter which contains number of version keys.
const I_T_BYTE* wrapKey	This is another Symmetric key through which key will be wrapped.
const I_T_AESKeyWrapFormat aesWrapFormat	The wrap format specifies the supported format in AES-Key wrap: I_T_ExportAESKeyWrapFormat_NONE : No Wrap Format is selected. I_T_ExportAESKeyWrapFormat_PEM_PKCS1 : Private key is exported in PKCS#1 format , public key pkcs#1 by default. I_T_ExportAESKeyWrapFormat_PEM_PKCS8 : Private key is exported in PKCS#8 format , public key pkcs#1 by default.
I_T_BYTE *** ppWrappedKeyBytes	On output, *ppWrappedKeyBytes will be assigned a pointer to the wrapped key bytes of the key. The memory pointed to is allocated by this function. Use the function I_C_Free to deallocate the memory.
unsigned int ** pWrappedKeyBytesLen	On output, *pWrappedKeyBytes will be assigned the number of key bytes of the key.
algo	The algorithem

Description

This function exports key bytes of a symmetric key in a wrapped form. The user must be the owner of the key or must have the permission to export the key. The wrap format specifies the algorithm used to wrap the plain key.

2.1.25 I_C_ExportCAChain Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ExportCAChain(I_O_Session sessionHandle, const I_T_CHAR *
caName, I_T_CHAR ** data, I_T_UINT * dataSize);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session sessionHandle	The current session.

const I_T_CHAR * caName	The name of the certificate for which CA chain is exported.
I_T_CHAR ** data	The output buffer that will receive the CA chain data. The function will allocate the memory. It should be freed by using I_C_Free when done.
I_T_UINT * dataSize	On input, this parameter specifies the size of the data buffer. On output, this parameter is set to the number of bytes actually written to the output data buffer.

Description

This function exports a CA chain from the DataSecure to a specified format.

2.1.26 I_C_ExportCertificate Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ExportCertificate(I_O_Session sessionHandle, const I_T_CHAR *
certificateName, I_T_ExportFormat exportFormat, const I_T_CHAR * password, I_T_CHAR
** data, I_T_UINT * dataSize);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session sessionHandle	The current session.
const I_T_CHAR * certificateName	The name of the certificate to export. This is a NULL-terminated character array.
I_T_ExportFormat exportFormat	The format of the exported certificate data. This must be a value from the I_T_Export_Format enum. **Note:** If exportFormat is I_T_ExportFormat_PEM_PKCS1_CERT_ONLY, but the certificateName is a certificate/key combination, only the certificate will be exported.
const I_T_CHAR * password	The password required when exporting to PKCS#12 format. This is a NULL-terminated character array. If exportFormat is I_T_ExportFormat_PKCS12, the password parameter must be set. Otherwise, this parameter must be NULL or an empty string.
I_T_CHAR ** data	The output buffer that will receive certificate data. The function will allocate the memory. It should be freed by using I_C_Free when done. **Note:** When exporting in PKCS#12 format, the output data will be binary.
I_T_UINT * dataSize	On input, this parameter specifies the size of the data buffer. On output, this parameter is set to the number of bytes actually written to the output data buffer.

Description

This function exports a certificate or a certificate/key combination from the DataSecure to a specified format. The certificate or certificate/key combination is in one of the PEM or PKCS formats defined in I_T_ExportFormat.

2.1.27 I_C_ExportKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ExportKey(I_O_Session sessionHandle, const I_T_CHAR *
KeyName, int KeyType, I_T_ExportFormat Format, I_T_CHAR ** KeyBytes, I_T_UINT *
KeyBytesLen);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session sessionHandle	The current session.
const I_T_CHAR * KeyName	Name of key to be exported.
int KeyType	Type of key to be exported. 1 for Public Key & 2 for private key.
I_T_ExportFormat Format	Format in which key will be exported. Applicable only for private key type. 1 for PEM-SEC1(default)& 2 for PEM-PKCS#8.
I_T_CHAR ** KeyBytes	[out] Character buffer to store exported key bytes.
I_T_UINT * KeyBytesLen	[out] Pointer to integer containing length of exported key bytes.

Description

This function Export Key according to key type and format specified by user

2.1.28 I_C_ExportPublicKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ExportPublicKey(I_O_Session session, const I_T_CHAR *
keyName, I_T_CHAR ** keyBytes);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * keyName	The name of the RSA key to be exported.

I_T_CHAR ** keyBytes	A pointer to the returned RSA public key. The memory pointed to is allocated by this function. Use the function I_C_Free to deallocate the memory.

Description

This function exports the public portion of an RSA key pair.

2.1.29 I_C_ExportSymmetricKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ExportSymmetricKey(I_O_Session session, const I_T_CHAR *
keyName, I_T_BYTE ** ppKeyBytes, I_T_UINT * pKeyBytesLen);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * keyName	The name of the key to be exported. The key must be exportable.
I_T_UINT * pKeyBytesLen	On output, *pKeyBytesLen will be assigned the number of key bytes of the key.
ppkeyBytes	On output, *ppKeyBytes will be assigned a pointer to the key bytes of the key. This must be stored. The memory pointed to is allocated by this function. Use the function I_C_Free to deallocate the memory.

Description

This function exports key bytes of a symmetric key. The user must be the owner of the key or must have permission to export the key.

2.1.30 I_C_ExportWrappedKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ExportWrappedKey(I_O_Session handle, const I_T_CHAR *
keyName, const I_T_BYTE* wrapPublicKey, const I_T_UINT wrapPublicKeyLen, const
I_T_KeyWrapFormat wrapFormat, I_T_BYTE ** ppWrappedKeyBytes, I_T_UINT *
pWrappedKeyBytesLen);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	The handle to the session.
const I_T_CHAR * keyName	The name of the symmetric key that should be exported. The key must be exportable.
const I_T_BYTE* wrapPublicKey	The PEM encoded RSA Public Key, which needs to be used for encrypting the key bytes with PKCS1v1.5,PKCS1v2.1/RSAOAEP-SHA256 , PKCS1v2.1/RSAOAEP-SHA384, PKCS1v2.1/RSAOAEP-SHA512 and PKCS1v2.1/RSAOAEP-SHA1
const I_T_UINT wrapPublicKeyLen	The length of Public Key buffer supplied by the caller.
const I_T_KeyWrapFormat wrapFormat	Decides how the key bytes are encrypted. I_T_ExportKeyWrapFormat_RAW_PKCS1v15 , I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA256 , I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA384 , I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA512 and I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA1 are the formats supported
I_T_BYTE ** ppWrappedKeyBytes	On output, *ppWrappedKeyBytes will be assigned a pointer to the wrapped key bytes of the key. The memory pointed to is allocated by this function. Use the function I_C_Free to deallocate the memory.
I_T_UINT * pWrappedKeyBytesLen	On output, *pWrappedKeyBytes will be assigned the number of key bytes of the key.

Description

This function exports key bytes of a symmetric key in a wrapped form. The user must be the owner of the key or must have the permission to export the key. The wrap format specifies the algorithm used to wrap the plain key.

I_T_ExportKeyWrapFormat_RAW_PKCS1v15 ,I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA256 , I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA384 , I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA512 are the wrap formats supported. When the format given is I_T_ExportKeyWrapFormat_RAW_PKCS1v15 , the key bytes are encrypted using RSA public key using PKCS1v1.5 format I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA256, key bytes are encrypted using RSA public key using PKSC1v2.1/RSAOAEP-SHA256 format. I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA384 , key bytes are encrypted using RSA public key using PKCS1v2.1/RSAOAEP-SHA384 format. I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA512 , key bytes are encrypted using RSA public key using PKCS1v2.1/RSAOAEP-SHA512 format.

2.1.31 I_C_FindInAttributeList Function

C/C++

```
I_T_RETURN FUNCEXP I_C_FindInAttributeList(I_O_AttributeList attributeList, const
I_T_CHAR * attributeName, I_T_CHAR ** ppAttributeValue);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList attributeList	An AttributeList object.
const I_T_CHAR * attributeName	The name of the attribute whose value is to be found. Both custom and system attribute lists may be passed.
I_T_CHAR ** ppAttributeValue	If attributeList contains an attribute with the attributeName, I_E_OK is returned and *ppAttributeValue contains a pointer to the value of the attribute. Otherwise, I_E_END is returned.

Description

This function finds an attribute value in an attribute list.

2.1.32 I_C_FindInAttributeListwithType Function

C/C++

```
I_T_RETURN FUNCEXP I_C_FindInAttributeListwithType(I_O_AttributeList attributeList,
const I_T_CHAR * attributeName, I_T_CHAR ** ppAttributeValue, I_T_INT *
attributeDataType);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList attributeList	An AttributeList object.
const I_T_CHAR * attributeName	The name of the attribute whose value is to be found. Both custom and system attribute lists may be passed.
I_T_CHAR ** ppAttributeValue	If attributeList contains an attribute with the attributeName, I_E_OK is returned and *ppAttributeValue contains a pointer to the value of the attribute.
I_T_INT * attributeDataType	If the attributeList constains an attribute with attributeName, I_E_OK is returned and attributeDataType constains the datatype of the attribute. Otherwise, I_E_END is returned.

Description

This function finds an attribute value and type in an attribute list.

2.1.33 I_C_FindInstanceInAttributeList Function

C/C++

```
I_T_RETURN FUNCEXP I_C_FindInstanceInAttributeList(I_O_AttributeList attributeList,
const I_T_CHAR * attributeName, I_T_CHAR ** ppAttributeValue, I_T_UINT
instanceNumber);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList attributeList	An AttributeList object. Only system attribute lists may be passed.
const I_T_CHAR * attributeName	The name of the attribute whose value for a specific instance is to be retrieved.
I_T_CHAR ** ppAttributeValue	If attributeList contains the specific instance of an attribute with the attributeName, I_E_OK is returned and *ppAttributeValue contains a pointer to the value of the attribute. Otherwise, I_E_END is returned. There are no "holes" - the lowest value of the parameter instance that causes an I_E_END represents one more than the number of instances that exist with the given attributeName.
I_T_UINT instanceNumber	The specific instance of the attribute with the attributeName whose value is to be retrieved. The instance numbering starts with 1 (not with zero!).

Description

This function retrieves the value of a specific instance of an attribute with the given attributeName. This function was designed to be used with an attributeList that may contain multiple instances of attributes with the same attributeName. Typically, it is used to retrieve the values of all instances of an attributeName.

2.1.34 I_C_FindKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_FindKey(I_O_Session session, const I_T_CHAR* pNameString,
const I_T_CHAR* pValueString, I_O_StringList* pKeyNamesList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR* pNameString	A string containing attribute's name.
const I_T_CHAR* pValueString	A string containing attribute's value. pNameString or(and) pValueString must be specified. If pNameString is 0 or empty a search is being done by attribute's value, if pValueString is 0 or empty - then by attribute's name.
I_O_StringList* pKeyNamesList	[out] A list of found keys' names.

Description

This function returns a list of keys having certain attiribute

This function allocates memory for pKeyNamesList, therefore the user must call later I_C_DeleteStringList(pStringList); to prevent a memory leak

2.1.35 I_C_GetCiphertextHeaderLength Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetCiphertextHeaderLength(I_O_Session session, I_O_CipherSpec cipher, const I_T_BYTE * cipherText, I_T_UINT cipherTextLen, I_T_UINT * cipherHeaderLen);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher spec.
const I_T_BYTE * cipherText	A pointer to a buffer containing tagged cipher text. Must not be NULL.
I_T_UINT cipherTextLen	The length of the cipherText. Must be greater than zero.
I_T_UINT * cipherHeaderLen	On output, cipherHeaderLen will be assigned the number of bytes consumed by the tag. Must not be NULL.

Description

This function returns the length of the ciphertext's header. You can use this function to get data about both versioned and non-versioned keys.

2.1.36 I_C_GetKeyAttributes Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetKeyAttributes(I_O_Session session, const I_T_CHAR *
keyName, I_O_AttributeList * pSystemAttributeList, I_O_AttributeList *
pCustomAttributeList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * keyName	The key whose attributes are to be retrieved.
I_O_AttributeList * pSystemAttributeList	On output, *pSystemAttributeList contains a pointer to an attribute list of the system attributes. The attribute names returned are:
I_O_AttributeList * pCustomAttributeList	On output, *pCustomAttributeList contains a pointer to an attribute list of the custom attributes.

Description

This function returns the attributes of a key. Key attributes include keysize, permissions, algorithm, deletable, exportable, and any user-defined elements.

Note: The user must be the owner of the key, or must have access granted to the key.

2.1.37 I_C_GetKeyNames Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetKeyNames(I_O_Session handle, I_T_KeyNameAttributes *
keyAttr, I_O_StringList* pKeyNamesList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_T_KeyNameAttributes * keyAttr	Structure containing attributes according to which keys will be searched
I_O_StringList* pKeyNamesList	[out] A list of found keys' names.
sessionHandle	The current session.

Description

This function get Key Names according to custom attributes listed or fingerprint

2.1.38 I_C_GetUserAttributes Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetUserAttributes(I_O_Session session, const I_T_CHAR *
username, I_O_AttributeList * pSystemAttributeList, I_O_AttributeList *
pCustomAttributeList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * username	The user whose attributes should be retrieved. If a null pointer is sent, then the attributes of the logged on user are retrieved. All users may retrieve their own attributes. Only users with administrative privileges may retrieve attributes of other users.
I_O_AttributeList * pSystemAttributeList	On output, *pSystemAttributeList contains a pointer to an AttributeList of the system attributes. The attribute names returned are:
I_O_AttributeList * pCustomAttributeList	On output, *pCustomAttributeList contains a pointer to an AttributeList of the custom attributes.

Description

This function returns the attributes of a user.

2.1.39 I_C_GetUserSpec Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetUserSpec(const I_T_USpecParam field, const void * value,
int * valueLen, I_O_UserSpec* userSpec);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
const I_T_USpecParam field	The Parameter name from enum I_T_USpecParam, For example, I_T_USPEC_AUTHTAG.
const void * value	The value corresponding to field eg: "SHA1".
int * valueLen	The valueLen corresponding to value passed.
userspec	The userspec will be passed. The pointer to an I_O_UserSpec to hold the returned object.

Description

This function get values from a UserSpec object.

A UserSpec defines additional User inputs. It may be reused in multiple cryptographic operations and may be used in more than one operation at a time.

2.1.40 I_C_ImportCertificate Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ImportCertificate(I_O_Session sessionHandle, const I_T_CHAR *
certificateName, I_T_BOOL deletableFlag, I_T_BOOL exportableFlag, I_O_GroupList
groupList, const I_T_CHAR * password, I_T_CHAR * data, I_T_UINT dataSize);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session sessionHandle	The current session.
const I_T_CHAR * certificateName	The name of the certificate to import.
I_T_BOOL deletableFlag	When true, this parameter allows the key to be deleted from the DataSecure.
I_T_BOOL exportableFlag	When true, this parameter allows the key to be exported from the DataSecure.
const I_T_CHAR * password	This password must be provided when the certificate is in PKCS#12 format. Otherwise, this parameter must be NULL or an empty string. This is a NULL-terminated character array.
I_T_CHAR * data	The certificate data.
I_T_UINT dataSize	The size of the imported certificate.
grouplist	This is the groupList that will access to the key. For more information, see I_C_CreateGroupListObject.

Description

This function imports a certificate or certificate/key combination into the DataSecure. The DataSecure will determine the certificate's format based on the data itself. If the certificate is in PKCS#12 format, you must provide the password.

2.1.41 I_C_ImportKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ImportKey(I_O_Session session, const I_T_CHAR * keyname,
I_T_BYTE * keyBytes, I_T_UINT keyBytesLen, I_O_KeyInfo keyinfo, I_O_GroupList
grouplist);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * keyname	The name of the new key.
I_T_BYTE * keyBytes	The key bytes to use for the new key.
I_T_UINT keyBytesLen	The length of the keyBytes array.
I_O_GroupList grouplist	A GroupList object. See I_C_CreateGroupListObject.
keyInfo	The keyInfo (algorithm name, key size, etc) for the key.

Description

This function imports a key to the server.

Note: The versioned keys cannot be imported.

2.1.42 I_C_Initialize Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Initialize(I_T_InitializationSource source, const I_T_CHAR *
path);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_T_InitializationSource source	This is the source of the initialization information. Possible values are an initialization file (Init_File), or an environment variable (Init_Environment). These values are discussed in "I_T_InitializationSourceEnum".
const I_T_CHAR * path	This is the path to the source. When source is Init_File, this is the path to the properties file. When source is Init_Environment, this is the environment variable that contains the location of the path.

Description

This function initializes the library.

2.1.43 I_C_LogEvent Function

C/C++

```
I_T_RETURN FUNCEXP I_C_LogEvent(I_O_Session session, const I_T_CHAR * logMessage);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * logMessage	A message to log on the server.

Description

This function logs a message on the server.

2.1.44 I_C_ModifyGroupPermissions Function

C/C++

```
I_T_RETURN FUNCEXP I_C_ModifyGroupPermissions(I_O_Session handle, const I_T_CHAR *
keyname, I_O_GroupList grouplist);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	The current session.
I_O_GroupList grouplist	A pointer to the groupList to be created.
keyName	The name of the key whose group permissions are to be modified

Description

This function modifies the permissions of a group for the specified key.

2.1.45 I_C_OpenSession Function

C/C++

```
I_T_RETURN FUNCEXP I_C_OpenSession(I_O_Session * session, I_T_AuthType authType,
const I_T_CHAR * username, const I_T_CHAR * authToken);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session * session	The session to be created.
I_T_AuthType authType	The session's authentication type.

const I_T_CHAR * username	The user name that will be used for the session.
const I_T_CHAR * authToken	The user's authentication information, such as a password.

Description

This function opens a new session. If KMIP is configured (by configuring the KMIP_IP parameter in the properties file), using I_C_OpenSession with the I_T_Auth_Password parameter returns failure. This happens because the Credential Base Object is not supported. You can use the I_T_AuthNoPassword parameter in this case.

2.1.46 I_C_OpenSessionPersistentCacheCallback Function

C/C++

```
I_T_RETURN FUNCEXP I_C_OpenSessionPersistentCacheCallback(I_O_Session * session,
I_T_AuthType authType, const I_T_CHAR * username, const I_T_CHAR * authToken,
I_C_PersistentCacheCallback callbackFunction);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session * session	The session to be created.
I_T_AuthType authType	The session's authentication type.
const I_T_CHAR * username	The user name that will be used for the session.
const I_T_CHAR * authToken	The user's authentication information, such as a password.
I_C_PersistentCacheCallback callbackFunction	A callback function to be called for persistent cache access.

Description

This function opens a new session supplying a persistent cache callback function.

2.1.47 I_C_OpenSessionPersistentCacheCallback_filepath Function

C/C++

```
I_T_RETURN FUNCEXP I_C_OpenSessionPersistentCacheCallback_filepath(I_O_Session *
session, I_T_AuthType authType, const I_T_CHAR * username, const I_T_CHAR *
authToken, I_C_PersistentCacheCallback callbackFunction, const I_T_CHAR * path);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session * session	The session to be created.
I_T_AuthType authType	The session's authentication type.
const I_T_CHAR * username	The user name that will be used for the session.
const I_T_CHAR * authToken	The user's authentication information, such as a password.
I_C_PersistentCacheCallback callbackFunction	A callback function to be called for persistent cache access.
const I_T_CHAR * path	filepath for properties file.

Description

This function opens a new session supplying a persistent cache callback function.

2.1.48 I_C_OpenSessionPersistentCacheParameters Function

C/C++

```
I_T_RETURN FUNCEXP I_C_OpenSessionPersistentCacheParameters(I_O_Session * session,
I_T_AuthType authType, const I_T_CHAR * username, const I_T_CHAR * authToken,
I_T_PersistentCacheParameters * pcParameters);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session * session	The session to be created.
I_T_AuthType authType	The session's authentication type.
const I_T_CHAR * username	The user name that will be used for the session.
const I_T_CHAR * authToken	The user's authentication information, such as a password.
I_T_PersistentCacheParameters * pcParameters	A pointer to the structure containing persistent cache parameters like thumbprint.

Description

This function opens a new session with a persistent cache certificate used for retrieving passphrase.

2.1.49 I_C_OpenSessionPersistentCachePassphrase Function

C/C++

```
I_T_RETURN FUNCEXP I_C_OpenSessionPersistentCachePassphrase(I_O_Session * session,
I_T_AuthType authType, const I_T_CHAR * username, const I_T_CHAR * authToken, const
I_T_BYTE * passphrase, const I_T_UINT passphraseLength);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session * session	The session to be created.
I_T_AuthType authType	The session's authentication type.
const I_T_CHAR * username	The user name that will be used for the session.
const I_T_CHAR * authToken	The user's authentication information, such as a password.
const I_T_BYTE * passphrase	A pointer to the passphrase.
const I_T_UINT passphraseLength	The length of the passphrase.

Description

This function opens a new session with a persistent cache passphrase.

2.1.50 I_C_OpenSessionPersistentCachePassphrase_filepath Function

C/C++

```
I_T_RETURN FUNCEXP I_C_OpenSessionPersistentCachePassphrase_filepath(I_O_Session *
session, I_T_AuthType authType, const I_T_CHAR * username, const I_T_CHAR *
authToken, const I_T_BYTE * passphrase, const I_T_UINT passphraseLength, const
I_T_CHAR * path);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session * session	The session to be created.
I_T_AuthType authType	The session's authentication type.
const I_T_CHAR * username	The user name that will be used for the session.
const I_T_CHAR * authToken	The user's authentication information, such as a password.
const I_T_BYTE * passphrase	A pointer to the passphrase.
const I_T_UINT passphraseLength	The length of the passphrase.
const I_T_CHAR * path	filepath for properties file.

Description

This function opens a new session with a persistent cache passphrase.

2.1.51 I_C_OpenSession_filepath Function

C/C++

```
I_T_RETURN FUNCEXP I_C_OpenSession_filepath(I_O_Session * session, I_T_AuthType
authType, const I_T_CHAR * username, const I_T_CHAR * authToken, const I_T_CHAR *
path);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session * session	The session to be created.
I_T_AuthType authType	The session's authentication type.
const I_T_CHAR * username	The user name that will be used for the session.
const I_T_CHAR * authToken	The user's authentication information, such as a password.
const I_T_CHAR * path	filepath for properties file.

Description

This function opens a new session. If KMIP is configured (by configuring the KMIP_IP parameter in the properties file), using I_C_OpenSession_filepath with the I_T_Auth_Password parameter returns failure. This happens because the Credential Base Object is not supported. You can use the I_T_AuthNoPassword parameter in this case.

2.1.52 I_C_RemoveFromAttributeList Function

C/C++

```
I_T_RETURN FUNCEXP I_C_RemoveFromAttributeList(I_O_AttributeList
customAttributeList, const I_T_CHAR * attributeName);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList customAttributeList	An attribute list object from which to remove an attribute. Note that only a custom attribute list may be passed.
const I_T_CHAR * attributeName	The name of the attribute that should be removed. All attribute instances with the name will be removed.

Description

This function removes an attribute from an I_O_AttributeList object.

2.1.53 I_C_SetKeyAttributes Function

C/C++

```
I_T_RETURN FUNCEXP I_C_SetKeyAttributes(I_O_Session session, const I_T_CHAR *
keyname, I_T_BOOL clearExistingAttributes, I_O_AttributeList customAttributeList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
const I_T_CHAR * keyname	The name of the key.
I_T_BOOL clearExistingAttributes	Removes existing attributes before setting the given attribute list. Setting this to false will have the passed customAttributeList merged with the existing values with any common names being overwritten.
I_O_AttributeList customAttributeList	The new attribute list.

Description

This function sets custom attributes for the key on the server. Only the owner of the key may modify the attributes.

2.1.54 I_C_SetKeyParameter Function

C/C++

```
I_T_RETURN FUNCEXP I_C_SetKeyParameter(I_O_Session session, const I_T_CHAR *
keyname, I_T_KeyParameterType keyParameterType, I_T_KeyParameterValue
keyParameterValue);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The session to be examined.
I_T_KeyParameterType keyParameterType	The parameter type being modified. Must be a member of the I_T_KeyParameterTypeEnum - either I_T_KeyLifecycleState to change the key state, or I_T_KeyVersion to create a new version. See the typedef enum for I_T_KeyParameterType for valid values.

I_T_KeyParameterValue keyParameterValue	The key parameter value. The new parameter value. Must be a member of the I_T_KeyParameterValueEnum. To change the key state, use I_T_KeyParameter_State_Active, I_T_KeyParameter_State_Restricted, or I_T_KeyParameter_State_Retired. To create a new key version, use I_T_KeyParameter_Version_Increment. See the typedef enum for I_T_KeyParameterValueEnum. Increments key versions, or alters key lifecycle states.
keyName	The name of the existing versioned key. When changing the key state, (when keyParameterType == I_T_KeyLifecycleState), keyName should be in the format, "key_name#number_to_alter". For example, given the key name for the versioned key, "SecureKey", and and the version to modify is, say, 3, keyName should be "SecureKey#3". When creating a new version (when keyParameterType == I_T_KeyVersion), keyName should be in the format, "key_name", without the "#" and version number.

Description

This function can modify the state of a key version or create a new version.

Note: This function only supports versioned keys. You cannot create a new version of a nonversioned key.

2.1.55 I_C_SetUserSpec Function

C/C++

```
I_T_RETURN FUNCEXP I_C_SetUserSpec(const I_T_USpecParam field, const void * value,
int valueLen, I_O_UserSpec* userSpec);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
const I_T_USpecParam field	The Parameter name from enum I_T_USpecParam, For example, I_T_USPEC_TWEAKALGO.
const void * value	The value corresponding to field eg: "SHA1".
int valueLen	The valueLen corresponding to value passed.
userspec	The userspec that will be created. The pointer to an I_O_UserSpec to hold the returned object.

Description

This function creates and Set values in a UserSpec object.

A UserSpec defines additional User inputs. It may be reused in multiple cryptographic operations and may be used in more than one operation at a time.

2.1.56 I_KC_AddToAttributeList Function

C/C++

```
I_KS_Result FUNCEXP I_KC_AddToAttributeList(I_KO_AttributeList attributeList, const
I_KS_Attribute* const attribute_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KO_AttributeList attributeList	[in] The handle to the Attribute List Opaque Object.
const I_KS_Attribute* const attribute_p	[in] A pointer to the Attribute structure, I_KS_Attribute, containing Attribute Name and Attribute Value to be added. The Attribute Value Type should match the attribute string specified in Attribute Name.

Returns

The result.

Description

This function adds attributes to the Attribute List. The added attributes will be sent in KMIP Request.

2.1.57 I_KC_Create Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Create(I_O_Session handle, I_KO_AttributeList
attributeList, const I_KT_ObjectType objectType);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.
const I_KT_ObjectType objectType	[in] The type of Managed Object to be created.

Returns

The result.

Description

This function creates the Symmetric Key Object with the KMIP server.

2.1.58 I_KC_CreateAttributeList Function

C/C++

```
I_KS_Result FUNCEXP I_KC_CreateAttributeList(I_KO_AttributeList * const
attributeList_p, const I_T_CHAR * const templateName_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KO_AttributeList * const attributeList_p	[out] A pointer to I_KO_AttributeList. The I_KO_AttributeList memory should be allocated by the caller. The returned attributeList_p object should not be used across different threads.
const I_T_CHAR * const templateName_p	[in] NULL or the template name, which can be used in the KMIP Request.

Returns

The result.

Description

This function creates a new Attribute List for use with KMIP Operations.

2.1.59 I_KC_Crypto Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Crypto(I_O_Session handle, I_KO_AttributeList
attributeList, int operation, const I_T_BYTE * InData, I_KS_Crypto_Response **const
crypto_response);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.
int operation	[in] A varaible to store operation(encrypt or decrypt).
const I_T_BYTE * InData	[in] A pointer to store input data by user, which will need to encrypt.

Description

This function will encrypt the data given by user

2.1.60 I_KC_FreeCryptoObject Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeCryptoObject(I_KS_Crypto_Response * crypto_response);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
handle	[in] The handle to the session object.
attributeList	[in,out] The handle to the Attribute List Opaque Object.
attributes	[in] A pointer to an array of Attribute Names whose values need to be fetched from the Key Management Server. A Value of (void **)0 will fetch all attributes.
attributeCount	[in] The count of Attribute Names present in the attributes array.

Returns

The result.

Description

This function retrieves the attributes of the Managed Object from the KMIP server.

2.1.61 I_KC_Get Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Get(I_O_Session handle, I_KO_AttributeList attributeList,
const I_KS_GetRequest *const getRequest_p, I_KS_Object **const object_pp);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.
const I_KS_GetRequest *const getRequest_p	[in] The structure containing the elements to be passed to Get Request. This parameter should be set by the user.
I_KS_Object **const object_pp	[out] A pointer to hold the Managed Object. Memory for storing the response Managed object is allocated by the library. It should be freed up by using I_KS_FreeManagedObject.

Returns

The result.

Description

This function retrieves a Managed Object from the Key Management Server.

2.1.62 I_KC_GetAttributes Function

C/C++

```
I_KS_Result FUNCEXP I_KC_GetAttributes(I_O_Session handle, I_KO_AttributeList
attributeList, const I_T_CHAR * const attributes[], const I_T_UINT attributeCount);
```

File

File: cadp_capi_kmipkeymgmt.h

Description

This is function I_KC_GetAttributes.

2.1.63 I_KC_GetResultReasonString Function

C/C++

```
I_T_PCCHAR FUNCEXP I_KC_GetResultReasonString(const I_KS_Result result);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
const I_KS_Result result	[in] The result obtained from the KMIP Key Management API.

Returns

The result reason string.

Description

This function returns the result reason.

2.1.64 I_KC_GetResultStatusString Function

C/C++

```
I_T_PCCHAR FUNCEXP I_KC_GetResultStatusString(const I_KS_Result result);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
const I_KS_Result result	[in] The result obtained from the KMIP Key Management API.

Returns

The result status string.

Description

This function returns the result status.

2.1.65 I_KC_GetWrappedKey Function

C/C++

```
I_KS_Result FUNCEXP I_KC_GetWrappedKey(I_O_Session handle, I_KO_AttributeList
attributeList, const I_KS_GetRequest *const getRequest_p, I_KS_Object **const
object_pp, I_T_CHAR * wrapping_key_UID);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.
const I_KS_GetRequest *const getRequest_p	[in] The structure containing the elements to be passed to Get Request. This parameter should be set by the user.
I_KS_Object **const object_pp	[out] A pointer to hold the Managed Object. Memory for storing the response Managed object is allocated by the library. It should be freed up by using I_KS_FreeManagedObject.
I_T_CHAR * wrapping_key_UID	[in] UID of key using which key bytes will be wrapped

Returns

The result.

Description

This function get wrapped key bytes for a Managed Object from the Key Management Server.

2.1.66 I_KC_Locate Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Locate(I_O_Session handle, const I_KT_StorageStatusMask
storageMask, I_KO_AttributeList attributeList, I_KS_UniqueIdentifiers** const
uniqueIdentifiers_pp, const I_T_UINT maxItems);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
const I_KT_StorageStatusMask storageMask	[in] This parameter specifies the storage mask of the objects to be searched for.
I_KO_AttributeList attributeList	[in] The handle to the Attribute List Opaque Object.
I_KS_UniqueIdentifiers** const uniqueIdentifiers_pp	[out] The parameter contains the returned Unique Identifiers.
const I_T_UINT maxItems	[in] The maximum number of object identifiers that the server can return. Specify 0 for 'no' limit.

Returns

The result.

Description

This function locates Managed Objects as per the specified search criteria.

2.1.67 I_KC_Query Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Query(I_O_Session handle, I_KT_QueryFunction
queryFunctions[], const I_T_UINT functionsCount, I_KS_QueryResponse** const
queryResponse_pp);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KT_QueryFunction queryFunctions[]	[in] An array holding the requested I_KT_QueryFunction values.
const I_T_UINT functionsCount	[in] The count of elements of the queryFunctions array.

I_KS_QueryResponse** const queryResponse_pp	[out] The response to the queries. The user needs to pass address of the pointer to I_KS_QueryResponse. The I_KT_QueryResponse memory will be allocated by the library.

Returns

The result.

Description

This function queries the server of its capabilites. I_C_FreeQueryResponse should be used to free up the memory allocated by the library, to hold the response. The queryFunctions parameter can be used to specify the requested capabilities. I_KT_QueryFunction_ApplicationNameSpaces response is not supported.

2.1.68 I_KC_ReKey Function

C/C++

```
I_KS_Result FUNCEXP I_KC_ReKey(I_O_Session handle, I_KO_AttributeList attributeList,
const I_KT_ObjectType objectType);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.
const I_KT_ObjectType objectType	[in] The type of Managed Object to be created.

Returns

The result.

Description

This function Recreate the Symmetric Key Object with the KMIP server.

2.1.69 I_KC_ReKeyPair Function

C/C++

```
I_KS_Result FUNCEXP I_KC_ReKeyPair(I_O_Session handle, I_KO_AttributeList attrComm,
I_KO_AttributeList attrPriv, I_KO_AttributeList attrPub, const I_KT_ObjectType
objectType);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attrComm	[in,out] The handle to the Commom template Attribute List Opaque Object.
I_KO_AttributeList attrPriv	[in,out] The handle to the Private Key Attribute List Opaque Object.
I_KO_AttributeList attrPub	[in,out] The handle to the Public Key Attribute List Opaque Object.
const I_KT_ObjectType objectType	[in] The type of Managed Object to be created.

Returns

The result.

Description

This function Recreate the Asymmetric Key Object with the KMIP server.

2.1.70 I_KC_Register Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Register(I_O_Session handle, I_KO_AttributeList
attributeList, const I_KS_Object * const object_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.
const I_KS_Object * const object_p	[in] A pointer to the Managed Object. This parameter contains the object to be registered with the Key Management Server. objectType_t should be set to match the object being set.

Returns

The result.

Description

This function registers the Managed Object with the KMIP server.

2.1.71 I_KC_RegisterAsymmetricKey Function

C/C++

```
I_KS_Result FUNCEXP I_KC_RegisterAsymmetricKey(I_O_Session handle,
I_KO_AttributeList pubKeyAttrList, const I_KS_Object *const pPubKey,
I_KO_AttributeList privKeyAttrList, const I_KS_Object *const pPrivKey);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList pubKeyAttrList	[in,out] The handle to the Attribute List Opaque Object for Public Key.
const I_KS_Object *const pPubKey	[in] A pointer to the Managed Object of Type Public Key. This parameter contains a public part of the Asymmetric Key to be registered with the Key Management Server.
I_KO_AttributeList privKeyAttrList	[in,out] The handle to the Attribute List Opaque Object for Private Key.
const I_KS_Object *const pPrivKey	[in] A pointer to the Managed Object of Type Private Key. This parameter contains a private part of the Asymmetric Key to be registered with the Key Management Server.

Returns

The result.

Description

This function registers the Managed Object of type Asymmestric Key with the KMIP server.

2.1.72 I_KC_RetrieveFromAttributeList Function

C/C++

```
I_KS_Result FUNCEXP I_KC_RetrieveFromAttributeList(I_KO_AttributeList attributeList,
const I_T_CHAR* const attributeName_p, I_KS_Attribute*** const attribute_ppp,
I_T_UINT* const valueCount);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KO_AttributeList attributeList	[in] The Attribute List Opaque Object.
const I_T_CHAR* const attributeName_p	[in] The name of the attribute in the string format.

I_KS_Attribute*** const attribute_ppp	[out] This parameter returns a pointer to an array of I_KS_Attribute pointers. The memory occupied by the pointer array should be freed up by the caller by using I_C_Free.
I_T_UINT* const valueCount	[out] This parameter returns the number of I_KS_AttributeValue pointers present in the array.

Returns

The result.

Description

This function retrieves attribute(s) from the Attribute List.

2.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	I_T_AESKeyWrapFormatEnum	The Wrap Format defines how the key bytes are encrypted. Use the I_T_AESKeyWrapFormat typedef in your code.
	I_T_Certificate_Usage	enum for CertificateUsage
	I_T_ConjunctiveOperator	enum for Conjunctive Operator
	I_T_Curve_ID	enum for EC keys
	I_T_AESKeyWrapFormat	The Wrap Format defines how the key bytes are encrypted. Use the I_T_AESKeyWrapFormat typedef in your code.
	I_T_CurveID	enum for EC keys

Structures

	Name	Description
	HKDF_KeyDetail	This is record HKDF_KeyDetail.
	I_KS_Crypto_Response	Structure containing response
	I_T_CertificateDetails	This is record I_T_CertificateDetails.
	I_T_KeyNameAttributes	This is record I_T_KeyNameAttributes.
	I_T_PersistentCacheParameters	This is record I_T_PersistentCacheParameters.
	I_T_keyDetails	This is record I_T_keyDetails.
	HKDF_KEYDETAIL	This is type HKDF_KEYDETAIL.
	I_T_KeyDetails	This is type I_T_KeyDetails.

2.2.1 HKDF_KeyDetail Structure

C/C++

```
struct HKDF_KeyDetail {
  I_T_CHAR derivationAlgo[12];
  I_T_CHAR* ikmKeyName;
  I_T_BYTE* salt;
  int salt_len;
  I_T_BYTE* info;
  int info_len;
};
```

File

File: cadp_capi_naekeymgmt.h

Description

This is record HKDF_KeyDetail.

2.2.2 I_KS_Crypto_Response Structure

C/C++

```
struct I_KS_Crypto_Response {
  I_KS_ByteString Encrypted_data;
  I_KS_ByteString IV_data;
};
```

File

File: kmipkeymgmttypes.h

Description

Structure containing response

2.2.3 I_T_AESKeyWrapFormatEnum Enumeration

C/C++

```
enum I_T_AESKeyWrapFormatEnum {
  I_T_ExportAESKeyWrapFormat_NONE = 0,
  I_T_ExportAESKeyWrapFormat_PEM_PKCS1 = 1,
  I_T_ExportAESKeyWrapFormat_PEM_PKCS8 = 2
};
```

File

File: cadp_capi_naekeymgmt.h

Members

Members	Description
I_T_ExportAESKeyWrapFormat_NONE = 0	No Wrap Format is selected.
I_T_ExportAESKeyWrapFormat_PEM_PKCS1 = 1	Private key is exported in PKCS#1 format , public key pkcs#1 by default
I_T_ExportAESKeyWrapFormat_PEM_PKCS8 = 2	Private key is exported in PKCS#8 format , public key pkcs#1 by default

Description

The Wrap Format defines how the key bytes are encrypted. Use the I_T_AESKeyWrapFormat typedef in your code.

2.2.4 I_T_CertificateDetails Structure

C/C++

```
struct I_T_CertificateDetails {
  I_T_CHAR * organizationName;
  I_T_CHAR * organizationalUnitName;
  I_T_CHAR * locality;
  I_T_CHAR * stateProvinceName;
  I_T_CHAR * countryName;
  I_T_CHAR * emailAddr;
  I_T_CHAR ** ipList;
  int ipLen;
  I_T_CHAR ** dnsList;
  int dnsLen;
};
```

File

File: cadp_capi_naekeymgmt.h

Description

This is record I_T_CertificateDetails.

2.2.5 I_T_Certificate_Usage Enumeration

C/C++

```
enum I_T_Certificate_Usage {
  I_T_Client,
  I_T_Server,
  I_T_IntermediateCA
};
```

File

File: cadp_capi_naekeymgmt.h

Description

enum for CertificateUsage

2.2.6 I_T_ConjunctiveOperator Enumeration

C/C++

```
enum I_T_ConjunctiveOperator {
  I_T_OR = 1,
  I_T_AND = 2
};
```

File

File: cadp_capi_naekeymgmt.h

Description

enum for Conjunctive Operator

2.2.7 I_T_Curve_ID Enumeration

C/C++

```
enum I_T_Curve_ID {
  I_T_secp224k1 = 1,
  I_T_secp224r1,
  I_T_secp256k1,
  I_T_secp384r1,
  I_T_secp521r1,
  I_T_prime256v1,
  I_T_brainpoolP224r1,
  I_T_brainpoolP224t1,
  I_T_brainpoolP256r1,
  I_T_brainpoolP256t1,
  I_T_brainpoolP384r1,
  I_T_brainpoolP384t1,
  I_T_brainpoolP512r1,
  I_T_brainpoolP512t1,
  invalidID = -1
};
```

File

File: cadp_capi_naekeymgmt.h

Description

enum for EC keys

2.2.8 I_T_KeyNameAttributes Structure

C/C++

```
struct I_T_KeyNameAttributes {
  I_T_ConjunctiveOperator conjunctiveOperator;
  I_T_UINT keyOffset;
```

```
  I_T_UINT maxKeys;
  I_O_AttributeList customAttributes;
  I_T_UINT keyCount;
  I_T_UINT totalKeyCount;
  I_T_CHAR * fingerprint;
};
```

File

File: cadp_capi_naekeymgmt.h

Description

This is record I_T_KeyNameAttributes.

2.2.9 I_T_PersistentCacheParameters Structure

C/C++

```
struct I_T_PersistentCacheParameters {
  I_T_BYTE * thumbprint;
};
```

File

File: cadp_capi.h

Description

This is record I_T_PersistentCacheParameters.

2.2.10 I_T_keyDetails Structure

C/C++

```
struct I_T_keyDetails {
  int curve_eq;
  HKDF_KEYDETAIL* hkdf_KeyDetails;
};
```

File

File: cadp_capi_naekeymgmt.h

Description

This is record I_T_keyDetails.

2.2.11 HKDF_KEYDETAIL Structure

C/C++

```
typedef struct HKDF_KeyDetail {
  I_T_CHAR derivationAlgo[12];
```

```
  I_T_CHAR* ikmKeyName;
  I_T_BYTE* salt;
  int salt_len;
  I_T_BYTE* info;
  int info_len;
} HKDF_KEYDETAIL;
```

File

File: cadp_capi_naekeymgmt.h

Description

This is type HKDF_KEYDETAIL.

2.2.12 I_T_AESKeyWrapFormat Enumeration

C/C++

```
typedef enum I_T_AESKeyWrapFormatEnum {
  I_T_ExportAESKeyWrapFormat_NONE = 0,
  I_T_ExportAESKeyWrapFormat_PEM_PKCS1 = 1,
  I_T_ExportAESKeyWrapFormat_PEM_PKCS8 = 2
} I_T_AESKeyWrapFormat;
```

File

File: cadp_capi_naekeymgmt.h

Members

Members	Description
I_T_ExportAESKeyWrapFormat_NONE = 0	No Wrap Format is selected.
I_T_ExportAESKeyWrapFormat_PEM_PKCS1 = 1	Private key is exported in PKCS#1 format , public key pkcs#1 by default
I_T_ExportAESKeyWrapFormat_PEM_PKCS8 = 2	Private key is exported in PKCS#8 format , public key pkcs#1 by default

Description

The Wrap Format defines how the key bytes are encrypted. Use the I_T_AESKeyWrapFormat typedef in your code.

2.2.13 I_T_CurveID Enumeration

C/C++

```
typedef enum I_T_Curve_ID {
  I_T_secp224k1 = 1,
  I_T_secp224r1,
  I_T_secp256k1,
  I_T_secp384r1,
  I_T_secp521r1,
  I_T_prime256v1,
  I_T_brainpoolP224r1,
  I_T_brainpoolP224t1,
```

```
    I_T_brainpoolP256r1,
    I_T_brainpoolP256t1,
    I_T_brainpoolP384r1,
    I_T_brainpoolP384t1,
    I_T_brainpoolP512r1,
    I_T_brainpoolP512t1,
    invalidID = -1
} I_T_CurveID;
```

File

File: cadp_capi_naekeymgmt.h

Description

enum for EC keys

2.2.14 I_T_KeyDetails Structure

C/C++

```
typedef struct I_T_keyDetails {
  int curve_eq;
  HKDF_KEYDETAIL* hkdf_KeyDetails;
} I_T_KeyDetails;
```

File

File: cadp_capi_naekeymgmt.h

Description

This is type I_T_KeyDetails.

2.3 Macros

The following table lists macros in this documentation.

Macros

Name	Description
AES_WRAP_FMT_PEM_PKCS1	AES Wrap formats for key export WRAP_FMT_[XYZ].
AES_WRAP_FMT_PEM_PKCS8	This is macro AES_WRAP_FMT_PEM_PKCS8.
CRYPTO_UTUL_H_	This is macro CRYPTO_UTUL_H_.
CS_ASCII	This enum used for validation of UTF mode
CS_UTF16BE	This is macro CS_UTF16BE.
CS_UTF16LE	This is macro CS_UTF16LE.
CS_UTF32BE	This is macro CS_UTF32BE.
CS_UTF32LE	This is macro CS_UTF32LE.
CS_UTF8	This is macro CS_UTF8.
DECRYPT	This is macro DECRYPT.

ENCRYPT	Identifier naming: I_KC_ denotes KMIP function. I_KS_ denotes KMIP structure object. I_KO_ denotes KMIP opaque object. I_KT_ denotes KMIP type or enum value.
I_T_LNG_ALG_EC	ECC
WRAP_FMT_PEM_PKCS1_LEN	AES Wrap format string length
WRAP_FMT_PEM_PKCS8_LEN	This is macro WRAP_FMT_PEM_PKCS8_LEN.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA1	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA1.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA1_LEN	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA1_LEN.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA256	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA256.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA256_LEN	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA256_LEN.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA384	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA384.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA384_LEN	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA384_LEN.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA512	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA512.
WRAP_FMT_PKCS1v21_RSAOAEP_SHA512_LEN	This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA512_LEN.

2.3.1 AES_WRAP_FMT_PEM_PKCS1 Macro

C/C++

```
#define AES_WRAP_FMT_PEM_PKCS1 "PEM-PKCS#1"
```

File

File: cadp_capi_naekeymgmt.h

Description

AES Wrap formats for key export WRAP_FMT_[XYZ].

2.3.2 AES_WRAP_FMT_PEM_PKCS8 Macro

C/C++

```
#define AES_WRAP_FMT_PEM_PKCS8 "PEM-PKCS#8"
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro AES_WRAP_FMT_PEM_PKCS8.

2.3.3 CRYPTO_UTUL_H_ Macro

C/C++

```
#define CRYPTO_UTUL_H_
```

File

File: CryptoDataUtility.h

Description

This is macro CRYPTO_UTUL_H_.

2.3.4 CS_ASCII Macro

C/C++

```
#define CS_ASCII 0
```

File

File: cadp_capi.h

Description

This enum used for validation of UTF mode

2.3.5 CS_UTF16BE Macro

C/C++

```
#define CS_UTF16BE 3
```

File

File: cadp_capi.h

Description

This is macro CS_UTF16BE.

2.3.6 CS_UTF16LE Macro

C/C++

```
#define CS_UTF16LE 2
```

File

File: cadp_capi.h

Description

This is macro CS_UTF16LE.

2.3.7 CS_UTF32BE Macro

C/C++

```
#define CS_UTF32BE 5
```

File

File: cadp_capi.h

Description

This is macro CS_UTF32BE.

2.3.8 CS_UTF32LE Macro

C/C++

```
#define CS_UTF32LE 4
```

File

File: cadp_capi.h

Description

This is macro CS_UTF32LE.

2.3.9 CS_UTF8 Macro

C/C++

```
#define CS_UTF8 1
```

File

File: cadp_capi.h

Description

This is macro CS_UTF8.

2.3.10 DECRYPT Macro

C/C++

```
#define DECRYPT 0
```

File

File: kmipkeymgmttypes.h

Description

This is macro DECRYPT.

2.3.11 ENCRYPT Macro

C/C++

```
#define ENCRYPT 1
```

File

File: kmipkeymgmttypes.h

Description

Identifier naming:

I_KC_ denotes KMIP function. I_KS_ denotes KMIP structure object. I_KO_ denotes KMIP opaque object. I_KT_ denotes KMIP type or enum value.

2.3.12 I_T_LNG_ALG_EC Macro

C/C++

```
#define I_T_LNG_ALG_EC "EC"
```

File

File: cadp_capi_crypto.h

Description

ECC

2.3.13 WRAP_FMT_PEM_PKCS1_LEN Macro

C/C++

```
#define WRAP_FMT_PEM_PKCS1_LEN 10
```

File

File: cadp_capi_naekeymgmt.h

Description

AES Wrap format string length

2.3.14 WRAP_FMT_PEM_PKCS8_LEN Macro

C/C++

```
#define WRAP_FMT_PEM_PKCS8_LEN 10
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PEM_PKCS8_LEN.

2.3.15 WRAP_FMT_PKCS1v21_RSAOAEP_SHA1 Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA1 "PKCS1v2.1/RSAOAEP-SHA1"
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA1.

2.3.16 WRAP_FMT_PKCS1v21_RSAOAEP_SHA1_LEN Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA1_LEN 22
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA1_LEN.

2.3.17 WRAP_FMT_PKCS1v21_RSAOAEP_SHA256 Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA256 "PKCS1v2.1/RSAOAEP-SHA256"
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA256.

2.3.18 WRAP_FMT_PKCS1v21_RSAOAEP_SHA256_LEN Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA256_LEN 24
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA256_LEN.

2.3.19 WRAP_FMT_PKCS1v21_RSAOAEP_SHA384 Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA384 "PKCS1v2.1/RSA0AEP-SHA384"
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA384.

2.3.20 WRAP_FMT_PKCS1v21_RSAOAEP_SHA384_LEN Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA384_LEN 24
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA384_LEN.

2.3.21 WRAP_FMT_PKCS1v21_RSAOAEP_SHA512 Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA512 "PKCS1v2.1/RSA0AEP-SHA512"
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA512.

2.3.22 WRAP_FMT_PKCS1v21_RSAOAEP_SHA512_LEN Macro

C/C++

```
#define WRAP_FMT_PKCS1v21_RSAOAEP_SHA512_LEN 24
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro WRAP_FMT_PKCS1v21_RSAOAEP_SHA512_LEN.

3 Library Management API

This chapter describes the Library Management API.

3.1 Library Management Functions

This section describes the Library Management functions available in CADP CAPI KMIP.

Functions

	Name	Description
	I_C_SetPassPhraseCallback	This function sets a callback function, I_C_PassphraseCallback, to collect the passphrase.
	I_C_CloseSession	This function closes a session.
	I_C_GetErrorString	This function returns the message for an error code. The function returns a NULL if the code is not in I_T_ErrorCodeEnum.
	I_C_GetLastError	This function returns the most recent error code for a session.
	I_C_Fini	This function closes the library. There are no parameters to pass.
	I_C_Free	This function deallocates the memory allocated by I_C_ExportSymmetricKey and I_C_ExportPublicKey.
	I_C_KeyRefresh	The Key Refresh feature provides a mechanism to update the Symmetric Key Cache in background. This feature helps in synchronizing keys with their states on DataSecure. The I_C_KeyRefresh() function synchronizes the keys present in Symmetric Cache with server. **Note:** The Key Refresh feature works only when the Symmetric_Key_Cache_Enabled parameter is set to yes or tcp_ok in the properties file. This means that this feature works only through the NAE XML protocol.

3.1.1 I_C_SetPassPhraseCallback Function

C/C++

```
I_T_RETURN FUNCEXP I_C_SetPassPhraseCallback(I_C_PassphraseCallback callback);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_C_PassphraseCallback callback	This is a function pointer, which is called to get the passphrase.

Description

This function sets a callback function, I_C_PassphraseCallback, to collect the passphrase.

3.1.2 I_C_CloseSession Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CloseSession(I_O_Session session);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session session	The session to be closed.

Description

This function closes a session.

3.1.3 I_C_GetErrorString Function

C/C++

```
I_T_PCCHAR FUNCEXP I_C_GetErrorString(I_T_RETURN errorCode);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_T_RETURN errorCode	The error code to be examined.

Description

This function returns the message for an error code.

The function returns a NULL if the code is not in I_T_ErrorCodeEnum.

3.1.4 I_C_GetLastError Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetLastError(I_O_Session session, I_T_RETURN * errorCode);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
I_O_Session session	The session to be examined.
I_T_RETURN * errorCode	The errorCode that will be returned.

Description

This function returns the most recent error code for a session.

3.1.5 I_C_Fini Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Fini();
```

File

File: cadp_capi.h

Description

This function closes the library. There are no parameters to pass.

3.1.6 I_C_Free Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Free(void * vp);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
void * vp	A void pointer. The memory that it points to will be deallocated. It deallocates the memory allocated by some CAPI functions.

Description

This function deallocates the memory allocated by I_C_ExportSymmetricKey and I_C_ExportPublicKey.

3.1.7 I_C_KeyRefresh Function

C/C++

```
I_T_RETURN FUNCEXP I_C_KeyRefresh(I_O_Session handle, I_O_CipherSpec cipher);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session handle	The current session.
I_O_CipherSpec cipher	The cipher containing the name of key.

Description

The Key Refresh feature provides a mechanism to update the Symmetric Key Cache in background. This feature helps in synchronizing keys with their states on DataSecure.

The I_C_KeyRefresh() function synchronizes the keys present in Symmetric Cache with server.

Note: The Key Refresh feature works only when the Symmetric_Key_Cache_Enabled parameter is set to yes or tcp_ok in the properties file. This means that this feature works only through the NAE XML protocol.

3.2 Library Management Typedefs

This section describes the Library Management typedefs available in CADP CAPI KMIP.

Types

Name	Description
I_C_PersistentCacheCallback	This typedef a callback function to be called for persistent cache access.
I_C_PassphraseCallback	This typedef defines a callback function that can be used to collect the passphrase string.
I_T_BOOL	This is the boolean type.
I_T_BYTE	Unsigned Char type. This typedef is platform independent.
I_T_CHAR	Char type. This typedef is platform independent.
I_T_INT	Int Type. On 64-bit architectures, this value is int 64.
I_T_INT32	32-bit Int.
I_T_INT64	64-bit Int.
I_T_PCCHAR	Pointer to constant character
I_T_RETURN	This is the return type.

I_T_UINT	Unsigned Long Int type. On 64-bit architectures, this value is unsigned int 64.
I_T_UINT32	32-bit Unsigned Int.
I_T_UINT64	64-bit Unsigned Int.

3.2.1 I_C_PersistentCacheCallback Type

C/C++

```
typedef int (* I_C_PersistentCacheCallback)(I_O_Session session, unsigned char *
const passphrase, unsigned int * const passphrase_len);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
session	The current session.
passphrase	The buffer area to copy the passphrase into.
passphrase_len	The size of the buffer allocated. Specify the passphrase length here.

Description

This typedef a callback function to be called for persistent cache access.

3.2.2 I_C_PassphraseCallback Type

C/C++

```
typedef char * (* I_C_PassphraseCallback)(void *arg);
```

File

File: cadp_capi.h

Parameters

Parameters	Description
arg	The Key_File path that is read from the properties file.

Description

This typedef defines a callback function that can be used to collect the passphrase string.

3.2.3 I_T_BOOL Type

C/C++

```
typedef I_T_UINT I_T_BOOL;
```

File

File: types.h

Description

This is the boolean type.

3.2.4 I_T_BYTE Type

C/C++

```
typedef unsigned char I_T_BYTE;
```

File

File: types.h

Description

Unsigned Char type. This typedef is platform independent.

3.2.5 I_T_CHAR Type

C/C++

```
typedef char I_T_CHAR;
```

File

File: types.h

Description

Char type. This typedef is platform independent.

3.2.6 I_T_INT Type

C/C++

```
typedef int I_T_INT;
```

File

File: types.h

Description

Int Type. On 64-bit architectures, this value is int 64.

3.2.7 I_T_INT32 Type

C/C++

```
typedef int32_t I_T_INT32;
```

File

File: types.h

Description

32-bit Int.

3.2.8 I_T_INT64 Type

C/C++

```
typedef int64_t I_T_INT64;
```

File

File: types.h

Description

64-bit Int.

3.2.9 I_T_PCCHAR Type

C/C++

```
typedef const char * I_T_PCCHAR;
```

File

File: types.h

Description

Pointer to constant character

3.2.10 I_T_RETURN Type

C/C++

```
typedef I_T_INT I_T_RETURN;
```

File

File: types.h

Description

This is the return type.

3.2.11 I_T_UINT Type

C/C++

```
typedef unsigned int I_T_UINT;
```

File

File: types.h

Description

Unsigned Long Int type. On 64-bit architectures, this value is unsigned int 64.

3.2.12 I_T_UINT32 Type

C/C++

```
typedef uint32_t I_T_UINT32;
```

File

File: types.h

Description

32-bit Unsigned Int.

3.2.13 I_T_UINT64 Type

C/C++

```
typedef uint64_t I_T_UINT64;
```

File

File: types.h

Description

64-bit Unsigned Int.

3.3 Library Management Enumerations

This section describes the Library Management enumerations available in CADP CAPI KMIP.

Enumerations

	Name	Description
	I_T_AuthTypeEnum	The authorization type describes how the session will be authenticated. Use the I_T_AuthType typedef in the code.
	I_T_ErrorCodeEnum	Error codes run numerically from 0 up, that is, are positive. **Do not use the numeric values in your code! Use the enum.**
	I_T_InitializationSourceEnum	This enum determines how the library is initialized. You can use the CADP_CAPI.properties file, or an environment variable that points to the properties file. Use the I_T_InitializationSource typdef in your code.

3.3.1 I_T_AuthTypeEnum Enumeration

C/C++

```
enum I_T_AuthTypeEnum {
  I_T_Auth_Password = 0,
  I_T_Auth_NoPassword = 1,
  I_T_Auth_KMIP = 2
};
```

File

File: cadp_capi.h

Members

Members	Description
I_T_Auth_Password = 0	The session will use Password Authentication.
I_T_Auth_NoPassword = 1	The session will not use Password Authentication. The supplied user name and password should be NULL while opening the session.
I_T_Auth_KMIP = 2	The session will use username & password for KMIP session authentication.

Description

The authorization type describes how the session will be authenticated. Use the I_T_AuthType typedef in the code.

3.3.2 I_T_ErrorCodeEnum Enumeration

C/C++

```
enum I_T_ErrorCodeEnum {
```

```
    I_E_OK = 0,
    I_E_ALREADY_INITIALIZED = 1,
    I_E_INVALID_INITIALIZE_PATH = 2,
    I_E_INVALID_INITIALIZE_SOURCE = 3,
    I_E_LIBRARY_UNINITIALIZED = 4,
    I_E_PROTOCOL_VERSION_MISMATCH = 5,
    I_E_PROPERTIES_SOURCE_NOT_FOUND = 6,
    I_E_PROPERTIES_FILE_NOT_FOUND = 7,
    I_E_NO_NAE_SERVERS = 8,
    I_E_CONF_FILE_VERSION_MISMATCH = 9,
    I_E_INVALID_PROPERTY = 10,
    I_E_CANNOT_WRITE_TO_LOG_FILE = 11,
    I_E_NAE_IP_PROPERTY_FORMAT_INVALID = 12,
    I_E_NO_MEM = 33,
    I_E_INVALID_PARAM = 34,
    I_E_INVALID_SESSION = 35,
    I_E_INVALID_OBJECT = 36,
    I_E_INVALID_AUTH_TYPE = 41,
    I_E_UNSUPPORTED_AUTH_TYPE = 42,
    I_E_LOGGED_OUT = 43,
    I_E_MISSING_AUTH_INFO = 44,
    I_E_SHORT_RANDOM = 51,
    I_E_KEY_NOT_EXPORTABLE = 52,
    I_E_SERVER_DOES_NOT_SUPPORT_BULK = 53,
    I_E_TOO_FEW_OPERATIONS_FOR_BULK = 54,
    I_E_INITIALIZATION_FAILED = 55,
    I_E_CRYPTO_BUFFER_TOO_SMALL = 56,
    I_E_UNSUPPORTED_ALGORITHM = 57,
    I_E_INCORRECT_KEY_TYPE = 58,
    I_E_ERROR_IN_BULK_OPERATION = 59,
    I_E_CONNECTION_FAILED = 70,
    I_E_SERVER_UNAVAILABLE = 71,
    I_E_SERVER_DOES_NOT_BATCH = 72,
    I_E_CONNECTION_READ_ERROR = 74,
    I_E_CONNECTION_WRITE_ERROR = 75,
    I_E_CONNECTION_ERROR = 76,
    I_E_CONNECTION_FATAL_ERROR = 77,
    I_E_CONNECTION_NONFATAL_ERROR = 78,
    I_E_CONNECTION_INVALID_RESPONSE = 79,
    I_E_CONNECTION_INVALID_DATA_SIZE = 80,
    I_E_CONNECTION_BUFFER_FULL = 81,
    I_E_CONNECTION_NO_MEM = 82,
    I_E_CONNECTION_INVALID_XML = 83,
    I_E_CONNECTION_INVALID = 84,
    I_E_UNKNOWN_SERVER_ERROR = 85,
    I_E_SERVER_AUTHENTICATION_FAILED = 86,
    I_E_SERVER_OPERATION_UNKNOWN = 87,
    I_E_INVALID_NAE_USER = 88,
    I_E_NAE_AUTHENTICATION_REQUIRED = 89,
    I_E_NAE_DATA_TOO_LONG = 90,
    I_E_DATA_SIZE_IS_NOT_BLOCK_SIZE_MULTIPLE = 91,
    I_E_INVALID_PADDING = 92,
    I_E_OPERATION_NOT_SUPPORTED = 93,
    I_E_INVALID_ALGORITHM_FOR_KEY = 94,
    I_E_INVALID_DATA_SIZE = 95,
    I_E_INVALID_KEY_NAME = 96,
    I_E_UNKNOWN_KEY = 97,
    I_E_COULD_NOT_INITIALIZE_KEY = 98,
    I_E_KEY_DELETED_OR_MODIFIED = 99,
    I_E_COULD_NOT_GENERATE_KEY = 100,
    I_E_KEY_ALREADY_EXISTS = 101,
```

```
    I_E_UNSUPPORTED_KEY_SIZE = 102,
    I_E_INVALID_KEY_SIZE = 103,
    I_E_INVALID_PERMISSIONS = 104,
    I_E_GLOBAL_KEY_CANNOT_HAVE_PERMISSIONS = 105,
    I_E_KEY_IS_NOT_DELETABLE = 106,
    I_E_INVALID_IV = 107,
    I_E_INVALID_IV_SIZE = 108,
    I_E_INVALID_ALGORITHM = 109,
    I_E_LOCAL_CRYPT_GENERIC_ERROR = 110,
    I_E_CANNOT_ENABLE_KEY_CACHE = 111,
    I_E_END = 112,
    I_E_INTERNAL_ERROR = 113,
    I_E_ALREADY_EXISTS = 114,
    I_E_TR_GROUP_REQUIRED = 115,
    I_E_SERVER_EXCEEDED_MAX_NUMBER_OF_KEYS = 116,
    I_E_INSUFFICIENT_PERMISSIONS = 117,
    I_E_GLOBAL_KEY_USAGE_NOT_ALLOWED = 118,
    I_E_KEY_CHANGE_OWNER_FAILED = 119,
    I_E_SET_KEY_META_DATA_FAILED = 120,
    I_E_MISSING_CREDENTIALS = 121,
    I_E_CANNOT_ENABLE_PERSISTENT_CACHE = 122,
    I_E_CANNOT_OBTAIN_PERSISTENT_CACHE_PASSPHRASE = 123,
    I_E_CANNOT_CONNECT_TO_PERSISTENT_CACHE = 124,
    I_E_CANNOT_READ_PERSISTENT_CACHE_ITEM = 125,
    I_E_CANNOT_WRITE_PERSISTENT_CACHE_ITEM = 126,
    I_E_USER_PASSPHRASE_CALLBACK_FAILED = 127,
    I_E_KEY_VERSION_DOES_NOT_EXIST = 128,
    I_E_NOT_VERSIONED_KEY = 129,
    I_E_USER_PASSPHRASE_OR_CALLBACK_NOT_SPECIFIED = 130,
    I_E_USER_PASSPHRASE_NOT_SET = 131,
    I_E_PERSISTENT_CACHING_DISALLOWED_BY_PROPERTIES_SETTINGS = 132,
    I_E_ERR_CONN_CAFILE = 133,
    I_E_ERR_CONN_CERTFILE = 134,
    I_E_ERR_CONN_PRIVATEKEY = 135,
    I_E_ERR_PERSISTENT_CACHE_REQUIRES_KEY_CACHE = 136,
    I_E_ERR_PERSISTENT_CACHE_DIRECTORY_NOT_SET = 137,
    I_E_ERR_SYMMETRIC_EXPIRY_GREATER_THAN_PERSISTENT_EXPIRY = 138,
    I_E_ERR_PERSISTENT_CACHE_EXPIRY_KEYS_VALUE_INVALID = 139,
    I_E_ERR_PERSISTENT_CACHE_MAX_SIZE_VALUE_INVALID = 140,
    I_E_INVALID_OPERATION = 141,
    I_E_INVALID_KEY_VERSION = 142,
    I_E_FAILED_TO_CREATE_VERSION = 143,
    I_E_KEY_VERSION_NOT_SUPPORTED_FOR_OPERATION = 144,
    I_E_INVALID_REQUEST_XML_FORMAT = 145,
    I_E_REQUEST_HEADER_TOO_LARGE = 146,
    I_E_INVALID_OR_MISSING_DATA_SECTION = 147,
    I_E_INVALID_DATA_CHUNK = 148,
    I_E_INVALID_CHUNK_SIZE = 149,
    I_E_DATA_TOO_LARGE_FOR_BULK = 150,
    I_E_AUTH_MISSING_ES_NAME = 151,
    I_E_INVALID_ESPROFILE = 152,
    I_E_INVALID_EDGESECURE = 153,
    I_E_MISMATCHED_EDGESECURE = 154,
    I_E_CIPHERTEXT_HEADER_DECODE_FAILED = 155,
    I_E_CIPHERTEXT_HEADER_ENCODE_FAILED = 156,
    I_E_OPERATION_DOES_NOT_SUPPORT_ALGORITHM = 157,
    I_E_CANNOT_OPEN_PERSISTENT_CACHE_FILE = 158,
    I_E_INVALID_OR_MISSING_MAC_VALUE = 159,
    I_E_KEY_IS_NOT_ASYMMETRIC = 160,
    I_E_INVALID_FORMAT = 161,
    I_E_PWD_FOR_PKCS12_ONLY = 162,
```

```
    I_E_PWD_REQUIRED_FOR_PKCS12 = 163,
    I_E_NOT_A_CERT = 164,
    I_E_CERT_HAS_NO_PRIV_KEY = 165,
    I_E_CERT_VERIFY_FAILED = 166,
    I_E_UNKNOWN_CA = 167,
    I_E_CA_EXPORT_FAILED = 168,
    I_E_CERT_CLONE_NOT_ALLOWED = 169,
    I_E_INVALID_OPERATION_ALGORITHM_PAIR = 170,
    I_E_FORMAT_NOT_NEEDED_FOR_SYMM_KEY = 171,
    I_E_INVALID_CONFIGURATION = 172,
    I_E_CONFIGURATION = -172,
    I_E_INVALID_TWEAK = -173,
    I_E_INVALID_FPE_FORMAT_TYPE = -174,
    I_E_NO_KEY_OBJECT_FOUND = 175,
    I_E_MISSING_PARAMETERS = -176,
    I_E_IV_NOT_SUPPORTED = -177,
    I_E_KMIP_SPEC_FILE = 201,
    I_E_CANNOT_DECRYPT_PERSISTENT_CACHE_FILE = 202,
    I_E_CANNOT_OBTAIN_CERTIFICATE_FOR_PERSISTENT_CACHE = 203,
    I_E_OPERATION_NOT_SUPPORTED_IN_LOCAL_MODE = 204,
    I_E_ERR_CONN_PROTOCOLMISMATCH = 205,
    I_E_KEY_NOT_FOUND = 1021,
    I_E_NODATA_READ = 10078,
    I_E_COULD_NOT_PARSE_REQUEST = 10101,
    I_E_NAE_COULD_NOT_IMPORT_KEY = 11421,
    I_E_NAE_INVALID_OR_MISSING_KEY_DATA = 11423,
    I_E_ENCRYPTED_TABLE_NOT_FOUND = 11600,
    I_E_ENCRYPTED_COLUMN_NOT_FOUND = 11601,
    I_E_SERVER_UNKNOWN_SERVER_ERROR = 20001,
    I_E_SERVER_OUT_OF_MEMORY = 20002,
    I_E_SERVER_NO_INSTALLED_LICENSES = 20040,
    I_E_SERVER_ALL_LICENSES_IN_USE = 20041,
    I_E_SERVER_PROTOCOL_MISMATCH = 20050,
    I_E_SERVER_UNRECOGNIZABLE_REQUEST = 20100,
    I_E_SERVER_UNSUPPORTED_PROTOCOL_VERSION = 20107,
    I_E_SERVER_INVALID_REQUEST_ID = 20108,
    I_E_SERVER_INVALID_VERSION_REQUEST = 20109,
    I_E_SERVER_REQUEST_NO_LONGER_SUPPORTED = 20110,
    I_E_SERVER_REQUEST_DISALLOWED = 20111,
    I_E_SERVER_KEY_EXPORT_REQUEST_DISALLOWED = 20112,
    I_E_SERVER_INVALID_KEYINFO_PARAMS = 20114,
    I_E_SERVER_INCOMPLETE_PERMISSIONS = 20115,
    I_E_SERVER_CANNOT_CLONE_VERSIONED_KEYS = 20116,
    I_E_SERVER_AUTH_MISSING_USER_NAME = 20200,
    I_E_SERVER_AUTH_MISSING_PASSWD = 20201,
    I_E_SERVER_AUTH_INVALID_USER_PASSWD = 20202,
    I_E_SERVER_INVALID_AUTH = 20203,
    I_E_SERVER_USER_CERT_MISMATCH = 20204,
    I_E_SERVER_AUTH_CANNOT_CONNECT = 20205,
    I_E_SERVER_AUTH_MISSING_CREDENTIALS = 20206,
    I_E_SERVER_AUTH_INVALID_TOKEN = 20207,
    I_E_SERVER_AUTH_TOKEN_EXPIRED = 20208,
    I_E_SERVER_UNAUTHORIZED_ACCESS = 20221,
    I_E_SERVER_PASSWORD_REQUIRED = 20230,
    I_E_SERVER_CLIENT_MUST_REAUTH = 20231,
    I_E_SERVER_CLIENT_CERTIFICATE_REQUIRED = 20240,
    I_E_SERVER_CLIENT_CERTIFICATE_INVALID = 20241,
    I_E_SERVER_CERT_IP_REQUIRED = 20242,
    I_E_SERVER_CERT_IP_INVALID = 20243,
    I_E_SERVER_CERT_IP_MISMATCH = 20244,
    I_E_SERVER_CLIENT_CIPHER_INVALID = 20245,
```

```
    I_E_SERVER_NEED_AUTH = 20264,
    I_E_SERVER_USER_CANNOT_ACCESS_SERVICE = 20280,
    I_E_SERVER_CRYPTO_FAILED = 20300,
    I_E_SERVER_CRYPTO_UPDATE_FAILED = 20304,
    I_E_SERVER_CRYPTO_FINAL_FAILED = 20305,
    I_E_SERVER_CRYPTO_OPERATION_FAILED = 20306,
    I_E_SERVER_CRYPTO_MAC_FAILED = 20307,
    I_E_SERVER_CRYPTO_MACV_FAILED = 20308,
    I_E_SERVER_CRYPTO_SIGN_FAILED = 20309,
    I_E_SERVER_CRYPTO_SIGNV_FAILED = 20310,
    I_E_SERVER_BASE64_ENCODE_FAILED = 20320,
    I_E_SERVER_CRYPTO_VERSION_MISMATCH = 20322,
    I_E_SERVER_KEY_CHANGE_OWNER_FAILED = 20400,
    I_E_SERVER_SET_KEY_META_DATA_FAILED = 20401,
    I_E_SERVER_DUPLICATE_CUSTOM_ATTRIBUTE = 20402,
    I_E_SERVER_INVALID_CUSTOM_ATTRIBUTE_VALUE = 20403,
    I_E_SERVER_MAX_NUM_CUSTOM_ATTRIBUTES_EXCEEDED = 20404,
    I_E_SERVER_MAX_CUSTOM_ATTRIBUTE_NAME_SIZE_EXCEEDED = 20405,
    I_E_SERVER_MAX_CUSTOM_ATTRIBUTE_VALUE_SIZE_EXCEEDED = 20406,
    I_E_SERVER_MAX_TOTAL_CUSTOM_ATTRIBUTE_SIZE_EXCEEDED = 20407,
    I_E_SERVER_INVALID_OWNER = 20408,
    I_E_SERVER_MAX_ACTIVE_VERSIONS_EXCEEDED = 20410,
    I_E_SERVER_INVALID_ATTRIBUTE_NAME = 20411,
    I_E_SERVER_CONFIGURATION_WRITE_FAILED = 21002,
    I_E_SERVER_INTERNAL_SERVER_COMM_TIMEOUT = 21003,
    I_E_SERVER_INTERNAL_SERVER_CONNECT_FAILED = 21004,
    I_E_SERVER_INVALID_PARAM_VALUE = 21100,
    I_E_SERVER_INVALID_OR_MISSING_OPERATIONS_BLOCK = 21300,
    I_E_SERVER_INVALID_OR_MISSING_OPERATION_NAME = 21301,
    I_E_SERVER_OPERATION_REQUIRES_ALGORITHM = 21303,
    I_E_SERVER_INVALID_ALGORITHM_OPERATION_PAIR = 21304,
    I_E_SERVER_INVALID_OR_MISSING_ALGORITHM_NAME = 21320,
    I_E_SERVER_UNKNOWN_ALGORITHM_NAME = 21321,
    I_E_SERVER_KEY_REQUIRED_FOR_THIS_ALGORITHM = 21322,
    I_E_SERVER_INVALID_KEY_ALGORITHM_PAIR = 21323,
    I_E_SERVER_NAE_CERT_NOT_VALID_FOR_CRYPTO = 21324,
    I_E_SERVER_INVALID_IV = 21340,
    I_E_SERVER_ALGORITHM_REQUIRES_IV = 21341,
    I_E_SERVER_ALGORITHM_DOES_NOT_REQUIRE_IV = 21342,
    I_E_SERVER_INVALID_OR_MISSING_SIGNATURE_VALUE = 21345,
    I_E_SERVER_INVALID_OR_MISSING_ES_NAME = 21370,
    I_E_SERVER_INVALID_OR_MISSING_DB_COLUMN_ID = 21371,
    I_E_SERVER_INVALID_OR_MISSING_KEY_NAME = 21400,
    I_E_SERVER_UNKNOWN_KEY_NAME = 21401,
    I_E_SERVER_NO_ACTIVE_VERSIONS = 21406,
    I_E_SERVER_PASSWORD_NOT_BASE_64 = 21407,
    I_E_SERVER_INVALID_KEY_STATE = 21429,
    I_E_SERVER_WEAK_DES_KEY = 21430,
    I_E_SERVER_REPL_PASSWORD_NOT_SET = 21442,
    I_E_SERVER_BOTH_WRAP_KEY_AND_WRAP_KEY_NAME = 21444,
    I_E_SERVER_INVALID_OR_MISSING_WRAP_ALGORITHM = 21445,
    I_E_SERVER_INVALID_OR_MISSING_WRAP_KEY_NAME = 21446,
    I_E_SERVER_KEY_IS_IN_USE_BY_PROFILES = 21451,
    I_E_SERVER_FAILED_TO_DELETE_KEY = 21452,
    I_E_SERVER_CERT_KEY_SIZE_NOT_ALLOWED = 21462,
    I_E_SERVER_RSA_ENCRYPTION_NOT_ALLOWED = 21463,
    I_E_SERVER_KEY_QUERY_EXCEEDED_NUM_KEYS = 21464,
    I_E_SERVER_INVALID_OR_MISSING_CRYPTO_PROFILE_NAME = 21470,
    I_E_SERVER_UNKNOWN_CRYPTO_PROFILE_NAME = 21471,
    I_E_SERVER_CERT_CAN_NOT_GENERATE_CR = 21500,
    I_E_SERVER_CERT_CAN_NOT_SIGN_CR = 21501,
```

```
    I_E_SERVER_CERT_CAN_NOT_INSTALL_CERT = 21502,
    I_E_SERVER_UNKNOWN_CERTIFICATE_REQUEST = 21510,
    I_E_SERVER_CERT_UNKNOWN_CERTIFICATE = 21511,
    I_E_SERVER_CERT_EXPORT_FAILED = 21512,
    I_E_SERVER_INVALID_OR_MISSING_CERT_NAME = 21520,
    I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE = 21521,
    I_E_SERVER_INVALID_OR_MISSING_COMMON_NAME = 21522,
    I_E_SERVER_INVALID_ORG_NAME = 21523,
    I_E_SERVER_INVALID_ORG_UNIT = 21524,
    I_E_SERVER_INVALID_LOC_NAME = 21525,
    I_E_SERVER_INVALID_STATE_NAME = 21526,
    I_E_SERVER_INVALID_OR_MISSING_COUNTRY_NAME = 21527,
    I_E_SERVER_INVALID_EMAIL = 21528,
    I_E_SERVER_INVALID_OR_MISSING_CA_NAME = 21529,
    I_E_SERVER_DUPLICATE_CERT_NAME = 21530,
    I_E_SERVER_INVALID_CERT_NAME = 21531,
    I_E_SERVER_CERT_NAME_TOO_LONG = 21532,
    I_E_SERVER_CERT_INVALID_KEY_SIZE = 21533,
    I_E_SERVER_CERT_COUNTRY_NAME_TOO_LONG = 21534,
    I_E_SERVER_CERT_COUNTRY_NAME_TOO_SHORT = 21534,
    I_E_SERVER_CERT_COMMON_NAME_TOO_LONG = 21535,
    I_E_SERVER_CERT_COMMON_NAME_TOO_SHORT = 21536,
    I_E_SERVER_CERT_EMAIL_TOO_LONG = 21537,
    I_E_SERVER_DUPLICATE_DN_FIELD = 21540,
    I_E_SERVER_INVALID_DN_FIELD_VALUE = 21541,
    I_E_SERVER_SUBJECT_WITHOUT_CERT = 21542,
    I_E_SERVER_CERT_WITHOUT_SUBJECT = 21543,
    I_E_SERVER_EXTENSIONS_WITHOUT_CERT = 21544,
    I_E_SERVER_EXTENSION_NOT_SUPPORTED = 21545,
    I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE_DATA_FORMAT = 21546,
    I_E_SERVER_NOT_A_CSR = 21547,
    I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE_USAGE = 21549,
    I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE_EXPIRY = 21550,
    I_E_SERVER_IS_A_CERT = 21551,
    I_E_SERVER_CA_CERT_DISALLOWED = 21553,
    I_E_SERVER_CERT_CRL_VERIFY_FAILED = 21554,
    I_E_SERVER_CERT_SIGN_INVALID = 21555,
    I_E_SERVER_CERT_HAS_EXPIRED = 21556,
    I_E_SERVER_CERT_IS_INACTIVE = 21557,
    I_E_SERVER_CERT_IS_INVALID = 21558,
    I_E_SERVER_INVALID_USER_NAME = 21601,
    I_E_SERVER_NAE_USER_ALREADY_EXISTS = 21602,
    I_E_SERVER_MISSING_PASSWD = 21620,
    I_E_SERVER_INVALID_PASSWORD = 21621,
    I_E_SERVER_WEAK_PASSWORD = 21622,
    I_E_SERVER_MISSING_GROUP_NAME = 21640,
    I_E_SERVER_INVALID_GROUP_NAME = 21641,
    I_E_SERVER_GROUP_ALREADY_EXISTS = 21642,
    I_E_SERVER_GROUP_DOES_NOT_EXIST = 21643,
    I_E_SERVER_USER_OWNS_KEY = 21660,
    I_E_SERVER_USER_IN_GROUP = 21661,
    I_E_SERVER_INVALID_USER_OP_LDAP = 21670,
    I_E_SERVER_USERMOD_INVALID_EDGESECURE = 21671,
    I_E_SERVER_SQL_MISSING_SQL = 21700,
    I_E_SERVER_SQL_MISSING_DATABASE_ALIAS = 21701,
    I_E_SERVER_SQL_BACKEND = 21702,
    I_E_SERVER_SQL_NO_MAPPING = 21703,
    I_E_SERVER_BACKEND_COMMUNICATION_FAILURE = 21800,
    I_E_SERVER_MISSING_MESSAGE_SIZE = 21900,
    I_E_SERVER_INVALID_MESSAGE_SIZE = 21901,
    I_E_SERVER_MISSING_MESSAGE = 21902,
```

```
    I_E_SERVER_EXCEEDED_MESSAGE_SIZE = 21903,
    I_E_ENUM_LENGTH
};
```

File

File: cadp_capi_err.h

Members

Members	Description
I_E_OK = 0	No error. Everything is OK.
I_E_ALREADY_INITIALIZED = 1	CADP CAPI library is already initialized.
I_E_INVALID_INITIALIZE_PATH = 2	Properties environment variable (IngrianNAE_Properties_Conf_File name) is empty.
I_E_INVALID_INITIALIZE_SOURCE = 3	I_T_InitializationSource passed with I_C_Initialize is invalid.
I_E_LIBRARY_UNINITIALIZED = 4	Library is not initialized.
I_E_PROTOCOL_VERSION_MISMATCH = 5	Server does not support any of the protocols that this client supports.
I_E_PROPERTIES_SOURCE_NOT_FOUND = 6	Cannot find a source for the CADP_CAPI.properties file.
I_E_PROPERTIES_FILE_NOT_FOUND = 7	Specified CADP_CAPI.properties was not found.
I_E_NO_NAE_SERVERS = 8	No NAE Servers were specified in the NAE_IP parameter.
I_E_CONF_FILE_VERSION_MISMATCH = 9	Unsupported version of the CADP_CAPI.properties file.
I_E_INVALID_PROPERTY = 10	At least one of the parameters in the CADP_CAPI.properties file is invalid.
I_E_CANNOT_WRITE_TO_LOG_FILE = 11	Cannot write to the log file specified in the Logfile property.
I_E_NAE_IP_PROPERTY_FORMAT_INVALID = 12	Illegal address in the NAE_IP parameter.
I_E_NO_MEM = 33	Out of memory.
I_E_INVALID_PARAM = 34	Invalid function parameter.
I_E_INVALID_SESSION = 35	Invalid session handle.
I_E_INVALID_OBJECT = 36	Invalid object handle.
I_E_INVALID_AUTH_TYPE = 41	Session authentication type is invalid. For this release, I_T_AuthType must be I_T_Auth_Password.
I_E_UNSUPPORTED_AUTH_TYPE = 42	Session authentication type is not supported.
I_E_LOGGED_OUT = 43	You cannot log back into session after logging out.

I_E_MISSING_AUTH_INFO = 44	Authentication argument is missing.
I_E_SHORT_RANDOM = 51	I_C_Random delivered fewer bytes than requested.
I_E_KEY_NOT_EXPORTABLE = 52	Unexportable key.
I_E_SERVER_DOES_NOT_SUPPORT_BULK = 53	NAE Server does not support bulk encryption.
I_E_TOO_FEW_OPERATIONS_FOR_BULK = 54	Bulk array is too small.
I_E_INITIALIZATION_FAILED = 55	Initialization failed.
I_E_CRYPTO_BUFFER_TOO_SMALL = 56	Output buffer is too small.
I_E_UNSUPPORTED_ALGORITHM = 57	Algorithm selected is not supported.
I_E_INCORRECT_KEY_TYPE = 58	Wrong key type used for the key import.
I_E_ERROR_IN_BULK_OPERATION = 59	BULK operation for all data elements did not succeed
I_E_CONNECTION_FAILED = 70	CADP CAPI cannot obtain a connection to a server.
I_E_SERVER_UNAVAILABLE = 71	Server is unavailable.
I_E_SERVER_DOES_NOT_BATCH = 72	Server does not support batching.
I_E_CONNECTION_READ_ERROR = 74	Server is unavailable: read error.
I_E_CONNECTION_WRITE_ERROR = 75	Server is unavailable: write error.
I_E_CONNECTION_ERROR = 76	Generic server error.
I_E_CONNECTION_FATAL_ERROR = 77	Fatal connection error.
I_E_CONNECTION_NONFATAL_ERROR = 78	Nonfatal connection error.
I_E_CONNECTION_INVALID_RESPONSE = 79	Connection error: invalid response.
I_E_CONNECTION_INVALID_DATA_SIZE = 80	Connection error: invalid data size.
I_E_CONNECTION_BUFFER_FULL = 81	Connection error: buffer full
I_E_CONNECTION_NO_MEM = 82	Connection error: out of memory.
I_E_CONNECTION_INVALID_XML = 83	Connection error: protocol error.
I_E_CONNECTION_INVALID = 84	Connection error: invalid command.
I_E_UNKNOWN_SERVER_ERROR = 85	Unknown server error.
I_E_SERVER_AUTHENTICATION_FAILED = 86	Authentication failed.
I_E_SERVER_OPERATION_UNKNOWN = 87	Unknown operation.
I_E_INVALID_NAE_USER = 88	Invalid username or password.
I_E_NAE_AUTHENTICATION_REQUIRED = 89	NAE user authentication is required.
I_E_NAE_DATA_TOO_LONG = 90	Data is too long for the operation.
I_E_DATA_SIZE_IS_NOT_BLOCK_SIZE_MULTIPLE = 91	Data is not a multiple of the cipher block size.
I_E_INVALID_PADDING = 92	Invalid ciphertext padding.
I_E_OPERATION_NOT_SUPPORTED = 93	Cryptographic operation is not supported.
I_E_INVALID_ALGORITHM_FOR_KEY = 94	Invalid algorithm for the key.

I_E_INVALID_DATA_SIZE = 95	Invalid data size.
I_E_INVALID_KEY_NAME = 96	Invalid key name.
I_E_UNKNOWN_KEY = 97	Unknown key.
I_E_COULD_NOT_INITIALIZE_KEY = 98	NAE server could not initialize the key.
I_E_KEY_DELETED_OR_MODIFIED = 99	Key was deleted or modified on the NAE Server.
I_E_COULD_NOT_GENERATE_KEY = 100	NAE Server could not generate the key.
I_E_KEY_ALREADY_EXISTS = 101	Key already exists on the NAE Server.
I_E_UNSUPPORTED_KEY_SIZE = 102	NAE server does not support this key size.
I_E_INVALID_KEY_SIZE = 103	Key size is invalid for this algorithm.
I_E_INVALID_PERMISSIONS = 104	Invalid key permissions.
I_E_GLOBAL_KEY_CANNOT_HAVE_PERMISSIONS = 105	Global keys cannot have permissions.
I_E_KEY_IS_NOT_DELETABLE = 106	Key is not deletable.
I_E_INVALID_IV = 107	IV is invalid.
I_E_INVALID_IV_SIZE = 108	IV size is invalid.
I_E_INVALID_ALGORITHM = 109	Algorithm is invalid.
I_E_LOCAL_CRYPT_GENERIC_ERROR = 110	Generic local cryptography error.
I_E_CANNOT_ENABLE_KEY_CACHE = 111	Cannot enable key cache.
I_E_END = 112	Reached end.
I_E_INTERNAL_ERROR = 113	Internal error - contact support.
I_E_ALREADY_EXISTS = 114	Already exists.
I_E_TR_GROUP_REQUIRED = 115	Group required.
I_E_SERVER_EXCEEDED_MAX_NUMBER_OF_KEYS = 116	Server exceeded max number of keys.
I_E_INSUFFICIENT_PERMISSIONS = 117	Insufficient permissions.
I_E_GLOBAL_KEY_USAGE_NOT_ALLOWED = 118	Key usage not allowed.
I_E_KEY_CHANGE_OWNER_FAILED = 119	Change owner failed.
I_E_SET_KEY_META_DATA_FAILED = 120	Meta data failed.
I_E_MISSING_CREDENTIALS = 121	Missing credentials.
I_E_CANNOT_ENABLE_PERSISTENT_CACHE = 122	Cannot enable persistent cache.
I_E_CANNOT_OBTAIN_PERSISTENT_CACHE_PASSPHRASE = 123	Cannot obtain persistent cache passphrase.
I_E_CANNOT_CONNECT_TO_PERSISTENT_CACHE = 124	Cannot connect to persistent cache.
I_E_CANNOT_READ_PERSISTENT_CACHE_ITEM = 125	Cannot read persistent cache item.
I_E_CANNOT_WRITE_PERSISTENT_CACHE_ITEM = 126	Cannot write persistent cache item.
I_E_USER_PASSPHRASE_CALLBACK_FAILED = 127	User passphrase callback failed.
I_E_KEY_VERSION_DOES_NOT_EXIST = 128	Key version does not exist.

I_E_NOT_VERSIONED_KEY = 129	Key is NOT a versioned key - #n disallowed.
I_E_USER_PASSPHRASE_OR_CALLBACK_NOT_SPECIFIED = 130	No user passphrase or callback function specified.
I_E_USER_PASSPHRASE_NOT_SET = 131	User callback function failed to set passphrase.
I_E_PERSISTENT_CACHING_DISALLOWED_BY_PROPERTIES_SETTINGS = 132	Persistent key caching disabled by properties settings.
I_E_ERR_CONN_CAFILE = 133	Connection CA file error.
I_E_ERR_CONN_CERTFILE = 134	Connection certificate file error.
I_E_ERR_CONN_PRIVATEKEY = 135	Connection private key error: Invalid entry for ?Key_File?, or invalid passphrase.
I_E_ERR_PERSISTENT_CACHE_REQUIRES_KEY_CACHE = 136	Persistent key caching requires symmetric/asymmetric key caching.
I_E_ERR_PERSISTENT_CACHE_DIRECTORY_NOT_SET = 137	Persistent_Cache_Directory is not set to a value.
I_E_ERR_SYMMETRIC_EXPIRY_GREATER_THAN_PERSISTENT_EXPIRY = 138	Symmetric_Key_Cache_Expiry > Persistent_Cache_Expiry_Keys. Persistent_Cache_Expiry_Keys must be greater.
I_E_ERR_PERSISTENT_CACHE_EXPIRY_KEYS_VALUE_INVALID = 139	Persistent_Cache_Expiry_Keys has an invalid value.
I_E_ERR_PERSISTENT_CACHE_MAX_SIZE_VALUE_INVALID = 140	Persistent_Cache_Max_Size has an invalid value.
I_E_INVALID_OPERATION = 141	Invalid operation.
I_E_INVALID_KEY_VERSION = 142	Invalid key version or none specified.
I_E_FAILED_TO_CREATE_VERSION = 143	Failed to create new key version.
I_E_KEY_VERSION_NOT_SUPPORTED_FOR_OPERATION = 144	Key version is not supported for this operation.
I_E_INVALID_REQUEST_XML_FORMAT = 145	Internal error: Client request has invalid XML format.
I_E_REQUEST_HEADER_TOO_LARGE = 146	Internal error: Header length cannot exceed 8192 bytes.
I_E_INVALID_OR_MISSING_DATA_SECTION = 147	Internal error: Invalid or missing data section.
I_E_INVALID_DATA_CHUNK = 148	Invalid data chunk.
I_E_INVALID_CHUNK_SIZE = 149	Invalid chunk size.
I_E_DATA_TOO_LARGE_FOR_BULK = 150	Data too large for bulk operations. The maximum size is 100 bytes.
I_E_AUTH_MISSING_ES_NAME = 151	Authorization missing ES name.
I_E_INVALID_ESPROFILE = 152	Invalid ES profile.
I_E_INVALID_EDGESECURE = 153	Invalid EdgeSecure.
I_E_MISMATCHED_EDGESECURE = 154	Mismatched EdgeSecure.
I_E_CIPHERTEXT_HEADER_DECODE_FAILED = 155	Ciphertext header decode failed.

I_E_CIPHERTEXT_HEADER_ENCODE_FAILED = 156	Ciphertext header encode failed.
I_E_OPERATION_DOES_NOT_SUPPORT_ALGORITHM = 157	Operation does not support this algorithm.
I_E_CANNOT_OPEN_PERSISTENT_CACHE_FILE = 158	Cannot open the persistent cache file.
I_E_INVALID_OR_MISSING_MAC_VALUE = 159	MAC value is invalid or missing.
I_E_KEY_IS_NOT_ASYMMETRIC = 160	Key is not asymmetric.
I_E_INVALID_FORMAT = 161	Invalid format.
I_E_PWD_FOR_PKCS12_ONLY = 162	Password for PKCS12 only.
I_E_PWD_REQUIRED_FOR_PKCS12 = 163	Password required for PKCS12.
I_E_NOT_A_CERT = 164	Not a certificate.
I_E_CERT_HAS_NO_PRIV_KEY = 165	Certificate does not have the private key.
I_E_CERT_VERIFY_FAILED = 166	Certificate verification failed.
I_E_UNKNOWN_CA = 167	Unknown CA.
I_E_CA_EXPORT_FAILED = 168	CA export failed.
I_E_CERT_CLONE_NOT_ALLOWED = 169	Certificate clone not allowed.
I_E_INVALID_OPERATION_ALGORITHM_PAIR = 170	Invalid operation algorithm pair.
I_E_FORMAT_NOT_NEEDED_FOR_SYMM_KEY = 171	Format not needed for the symmetric key.
I_E_INVALID_CONFIGURATION = 172	Invalid configuration.
I_E_INVALID_TWEAK = -173	Server error: invalid Tweak .
I_E_INVALID_FPE_FORMAT_TYPE = -174	Invalid FPE Format Type
I_E_NO_KEY_OBJECT_FOUND = 175	No Key object found on server
I_E_MISSING_PARAMETERS = -176	error : Missing Parameters
I_E_IV_NOT_SUPPORTED = -177	error : IV not supported
I_E_KMIP_SPEC_FILE = 201	Invalid KMIP Spec file specified in properties file
I_E_CANNOT_DECRYPT_PERSISTENT_CACHE_FILE = 202	Unable to decrypt persistant cache file
I_E_OPERATION_NOT_SUPPORTED_IN_LOCAL_MODE = 204	Unable to perform given operation in local mode
I_E_ERR_CONN_PROTOCOLMISMATCH = 205	TCP/SSL protocol mismatch
I_E_KEY_NOT_FOUND = 1021	No key found with this custom attribute
I_E_NODATA_READ = 10078	No data read.
I_E_COULD_NOT_PARSE_REQUEST = 10101	Could not parse the request.
I_E_NAE_COULD_NOT_IMPORT_KEY = 11421	NAE unable to import key.
I_E_NAE_INVALID_OR_MISSING_KEY_DATA = 11423	NAE error: key data invalid or missing.
I_E_ENCRYPTED_TABLE_NOT_FOUND = 11600	Encrypted table not found.
I_E_ENCRYPTED_COLUMN_NOT_FOUND = 11601	Encrypted column not found.
I_E_SERVER_UNKNOWN_SERVER_ERROR = 20001	Unknown server error.
I_E_SERVER_OUT_OF_MEMORY = 20002	Server error: out of memory.
I_E_SERVER_NO_INSTALLED_LICENSES = 20040	Server error: no licenses installed.

I_E_SERVER_ALL_LICENSES_IN_USE = 20041	Server error: all licenses in use.
I_E_SERVER_PROTOCOL_MISMATCH = 20050	Server error: protocol mismatch.
I_E_SERVER_UNRECOGNIZABLE_REQUEST = 20100	Server error: unrecognizable request.
I_E_SERVER_UNSUPPORTED_PROTOCOL_VERSION = 20107	Server error: unsupported protocol version.
I_E_SERVER_INVALID_REQUEST_ID = 20108	Server error: invalid request ID.
I_E_SERVER_INVALID_VERSION_REQUEST = 20109	Server error: invalid version request.
I_E_SERVER_REQUEST_NO_LONGER_SUPPORTED = 20110	Server error: request no longer supported.
I_E_SERVER_REQUEST_DISALLOWED = 20111	Server error: request not allowed.
I_E_SERVER_KEY_EXPORT_REQUEST_DISALLOWED = 20112	Server error: key export request not allowed.
I_E_SERVER_INVALID_KEYINFO_PARAMS = 20114	Server error: invalid keyinfo parameters.
I_E_SERVER_INCOMPLETE_PERMISSIONS = 20115	Server error: incomplete permissions.
I_E_SERVER_CANNOT_CLONE_VERSIONED_KEYS = 20116	Server error: cannot clone versioned keys.
I_E_SERVER_AUTH_MISSING_USER_NAME = 20200	Server authorization error: user name missing.
I_E_SERVER_AUTH_MISSING_PASSWD = 20201	Server authorization error: password missing.
I_E_SERVER_AUTH_INVALID_USER_PASSWD = 20202	Server authorization error: invalid user password.
I_E_SERVER_INVALID_AUTH = 20203	Server error: invalid authorization.
I_E_SERVER_USER_CERT_MISMATCH = 20204	Server error: user certificate mismatch.
I_E_SERVER_AUTH_CANNOT_CONNECT = 20205	Server authorization error: cannot connect.
I_E_SERVER_AUTH_MISSING_CREDENTIALS = 20206	Server authorization error: credentials missing.
I_E_SERVER_AUTH_INVALID_TOKEN = 20207	Server authorization error: invalid token.
I_E_SERVER_AUTH_TOKEN_EXPIRED = 20208	Server authorization error: token expired.
I_E_SERVER_UNAUTHORIZED_ACCESS = 20221	Server error: unauthorized access.
I_E_SERVER_PASSWORD_REQUIRED = 20230	Server error: password required.
I_E_SERVER_CLIENT_MUST_REAUTH = 20231	Server error: client must be reauthorized.
I_E_SERVER_CLIENT_CERTIFICATE_REQUIRED = 20240	Server error: client certificate required.
I_E_SERVER_CLIENT_CERTIFICATE_INVALID = 20241	Server error: client certificate invalid.
I_E_SERVER_CERT_IP_REQUIRED = 20242	Server error: Certificate IP required.

I_E_SERVER_CERT_IP_INVALID = 20243	Server error: Certificate IP invalid.
I_E_SERVER_CERT_IP_MISMATCH = 20244	Server error: Certificate IP mismatch.
I_E_SERVER_CLIENT_CIPHER_INVALID = 20245	Server error: Client cipher invalid.
I_E_SERVER_NEED_AUTH = 20264	Server error: authentication needed.
I_E_SERVER_USER_CANNOT_ACCESS_SERVICE = 20280	Server error: user cannot access service.
I_E_SERVER_CRYPTO_FAILED = 20300	Server error: crypto failed.
I_E_SERVER_CRYPTO_UPDATE_FAILED = 20304	Server error: crypto update failed.
I_E_SERVER_CRYPTO_FINAL_FAILED = 20305	Server error: crypto final failed.
I_E_SERVER_CRYPTO_OPERATION_FAILED = 20306	Server error: crypto operation failed.
I_E_SERVER_CRYPTO_MAC_FAILED = 20307	Server error: crypto MAC failed.
I_E_SERVER_CRYPTO_MACV_FAILED = 20308	Server error: crypto MAC verify failed.
I_E_SERVER_CRYPTO_SIGN_FAILED = 20309	Server error: crypto sign failed.
I_E_SERVER_CRYPTO_SIGNV_FAILED = 20310	Server error: crypto sign verify failed.
I_E_SERVER_BASE64_ENCODE_FAILED = 20320	Server error: Base64 encode failed.
I_E_SERVER_CRYPTO_VERSION_MISMATCH = 20322	Server error: crypto version mismatched.
I_E_SERVER_KEY_CHANGE_OWNER_FAILED = 20400	Server error: change key owner failed.
I_E_SERVER_SET_KEY_META_DATA_FAILED = 20401	Server error: set key metadata failed.
I_E_SERVER_DUPLICATE_CUSTOM_ATTRIBUTE = 20402	Server error: duplicate custom attribute.
I_E_SERVER_INVALID_CUSTOM_ATTRIBUTE_VALUE = 20403	Server error: invalid custom attribute value.
I_E_SERVER_MAX_NUM_CUSTOM_ATTRIBUTES_EXCEEDED = 20404	Server error: custom attributes exceeded maximum number.
I_E_SERVER_MAX_CUSTOM_ATTRIBUTE_NAME_SIZE_EXCEEDED = 20405	Server error: custom attribute name exceeded maximum size.
I_E_SERVER_MAX_CUSTOM_ATTRIBUTE_VALUE_SIZE_EXCEEDED = 20406	Server error: custom attribute value exceeded maximum size.
I_E_SERVER_MAX_TOTAL_CUSTOM_ATTRIBUTE_SIZE_EXCEEDED = 20407	Server error: total attribute size exceeded maximum size.
I_E_SERVER_INVALID_OWNER = 20408	Server error: invalid owner.
I_E_SERVER_MAX_ACTIVE_VERSIONS_EXCEEDED = 20410	Server error: active versions exceeded maximum value.
I_E_SERVER_INVALID_ATTRIBUTE_NAME = 20411	Server error: invalid attribute name.
I_E_SERVER_CONFIGURATION_WRITE_FAILED = 21002	Server error: configuration write failed.

I_E_SERVER_INTERNAL_SERVER_COMM_TIMEOUT = 21003	Server error: internal server communication timeout.
I_E_SERVER_INTERNAL_SERVER_CONNECT_FAILED = 21004	Server error: internal server connection failed.
I_E_SERVER_INVALID_PARAM_VALUE = 21100	Server error: invalid parameter value.
I_E_SERVER_INVALID_OR_MISSING_OPERATIONS_BLOCK = 21300	Server error: operations block is invalid or missing.
I_E_SERVER_INVALID_OR_MISSING_OPERATION_NAME = 21301	Server error: operation name is invalid or missing.
I_E_SERVER_OPERATION_REQUIRES_ALGORITHM = 21303	Server error: this operation requires algorithm.
I_E_SERVER_INVALID_ALGORITHM_OPERATION_PAIR = 21304	Server error: operation pair for the algorithm is invalid.
I_E_SERVER_INVALID_OR_MISSING_ALGORITHM_NAME = 21320	Server error: algorithm name is invalid or missing.
I_E_SERVER_UNKNOWN_ALGORITHM_NAME = 21321	Server error: unknown algorithm name.
I_E_SERVER_KEY_REQUIRED_FOR_THIS_ALGORITHM = 21322	Server error: key is required for this algorithm.
I_E_SERVER_INVALID_KEY_ALGORITHM_PAIR = 21323	Server error: algorithm pair for this key is invalid.
I_E_SERVER_NAE_CERT_NOT_VALID_FOR_CRYPTO = 21324	Server error: NAE certificate is invalid for crypto.
I_E_SERVER_INVALID_IV = 21340	Server error: invalid IV.
I_E_SERVER_ALGORITHM_REQUIRES_IV = 21341	Server error: algorithm requires IV.
I_E_SERVER_ALGORITHM_DOES_NOT_REQUIRE_IV = 21342	Server error: algorithm does not require IV.
I_E_SERVER_INVALID_OR_MISSING_SIGNATURE_VALUE = 21345	Server error: invalid or missing signature value.
I_E_SERVER_INVALID_OR_MISSING_ES_NAME = 21370	Server error: invalid or missing ES name.
I_E_SERVER_INVALID_OR_MISSING_DB_COLUMN_ID = 21371	Server error: invalid or missing DB column ID.
I_E_SERVER_INVALID_OR_MISSING_KEY_NAME = 21400	Server error: invalid or missing key name.
I_E_SERVER_UNKNOWN_KEY_NAME = 21401	Server error: unknown key name.
I_E_SERVER_NO_ACTIVE_VERSIONS = 21406	Server error: no active versions.
I_E_SERVER_PASSWORD_NOT_BASE_64 = 21407	Server error: password is not base64.
I_E_SERVER_INVALID_KEY_STATE = 21429	Server error: invalid key state.
I_E_SERVER_WEAK_DES_KEY = 21430	Server error: weak DES key.
I_E_SERVER_REPL_PASSWORD_NOT_SET = 21442	Server error: REPL PASSWORD NOT SET.
I_E_SERVER_BOTH_WRAP_KEY_AND_WRAP_KEY_NAME = 21444	Server error: BOTH WRAP KEY AND WRAP KEY NAME.
I_E_SERVER_INVALID_OR_MISSING_WRAP_ALGORITHM = 21445	Server error: wrap algorithm is invalid or missing.

I_E_SERVER_INVALID_OR_MISSING_WRAP_KEY_NAME = 21446	Server error: wrap key name is invalid or missing.
I_E_SERVER_KEY_IS_IN_USE_BY_PROFILES = 21451	Server error: key is in use by profiles.
I_E_SERVER_FAILED_TO_DELETE_KEY = 21452	Server error: failed to delete key.
I_E_SERVER_CERT_KEY_SIZE_NOT_ALLOWED = 21462	Server error: certificate key size not allowed.
I_E_SERVER_RSA_ENCRYPTION_NOT_ALLOWED = 21463	Server error: RSA encryption not allowed.
I_E_SERVER_KEY_QUERY_EXCEEDED_NUM_KEYS = 21464	Server error: KEY QUERY EXCEEDED NUM KEYS.
I_E_SERVER_INVALID_OR_MISSING_CRYPTO_PROFILE_NAME = 21470	Server error: crypto profile name is invalid or missing.
I_E_SERVER_UNKNOWN_CRYPTO_PROFILE_NAME = 21471	Server error: unknown crypto profile name.
I_E_SERVER_CERT_CAN_NOT_GENERATE_CR = 21500	Server error: certificate cannot generate CR.
I_E_SERVER_CERT_CAN_NOT_SIGN_CR = 21501	Server error: certificate cannot sign CR.
I_E_SERVER_CERT_CAN_NOT_INSTALL_CERT = 21502	Server error: certificate cannot install certificate.
I_E_SERVER_UNKNOWN_CERTIFICATE_REQUEST = 21510	Server error: unknown certificate request.
I_E_SERVER_CERT_UNKNOWN_CERTIFICATE = 21511	Server error: unknown certificate.
I_E_SERVER_CERT_EXPORT_FAILED = 21512	Server error: certificate export failed.
I_E_SERVER_INVALID_OR_MISSING_CERT_NAME = 21520	Server error: certificate name is invalid or missing.
I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE = 21521	Server error: certificate is invalid or missing.
I_E_SERVER_INVALID_OR_MISSING_COMMON_NAME = 21522	Server error: common name is invalid or missing.
I_E_SERVER_INVALID_ORG_NAME = 21523	Server error: organization name is invalid.
I_E_SERVER_INVALID_ORG_UNIT = 21524	Server error: organization unit is invalid.
I_E_SERVER_INVALID_LOC_NAME = 21525	Server error: location name is invalid.
I_E_SERVER_INVALID_STATE_NAME = 21526	Server error: state name is invalid.
I_E_SERVER_INVALID_OR_MISSING_COUNTRY_NAME = 21527	Server error: country name is invalid or missing.
I_E_SERVER_INVALID_EMAIL = 21528	Server error: invalid e-mail.
I_E_SERVER_INVALID_OR_MISSING_CA_NAME = 21529	Server error: CA name is invalid or missing.
I_E_SERVER_DUPLICATE_CERT_NAME = 21530	Server error: duplicate certificate name.
I_E_SERVER_INVALID_CERT_NAME = 21531	Server error: invalid certificate name.

I_E_SERVER_CERT_NAME_TOO_LONG = 21532	Server error: certificate name too long.
I_E_SERVER_CERT_INVALID_KEY_SIZE = 21533	Server error: invalid key size for the certificate.
I_E_SERVER_CERT_COUNTRY_NAME_TOO_LONG = 21534	Server error: certificate country name too long.
I_E_SERVER_CERT_COUNTRY_NAME_TOO_SHORT = 21534	Server error: certificate country name too short.
I_E_SERVER_CERT_COMMON_NAME_TOO_LONG = 21535	Server error: certificate common name too long.
I_E_SERVER_CERT_COMMON_NAME_TOO_SHORT = 21536	Server error: certificate common name too short.
I_E_SERVER_CERT_EMAIL_TOO_LONG = 21537	Server error: certificate e-mail too long.
I_E_SERVER_DUPLICATE_DN_FIELD = 21540	Server error: duplicate DN field.
I_E_SERVER_INVALID_DN_FIELD_VALUE = 21541	Server error: invalid DN field value.
I_E_SERVER_SUBJECT_WITHOUT_CERT = 21542	Server error: subject without certificate.
I_E_SERVER_CERT_WITHOUT_SUBJECT = 21543	Server error: certificate without subject.
I_E_SERVER_EXTENSIONS_WITHOUT_CERT = 21544	Server error: extension without certificate.
I_E_SERVER_EXTENSION_NOT_SUPPORTED = 21545	Server error: extension not supported.
I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE_DATA_FORMAT = 21546	Server error: certificate data format is inavlid or missing.
I_E_SERVER_NOT_A_CSR = 21547	Server error: not a CSR.
I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE_USAGE = 21549	Server error: certificate usage is inavlid or missing.
I_E_SERVER_INVALID_OR_MISSING_CERTIFICATE_EXPIRY = 21550	Server error: certificate expiry is inavlid or missing.
I_E_SERVER_IS_A_CERT = 21551	Server error: this is a certificate.
I_E_SERVER_CA_CERT_DISALLOWED = 21553	Server error: CA certificate not allowed.
I_E_SERVER_CERT_CRL_VERIFY_FAILED = 21554	Server error: certificate CRL verification failed.
I_E_SERVER_CERT_SIGN_INVALID = 21555	Server error: certificate sign invalid.
I_E_SERVER_CERT_HAS_EXPIRED = 21556	Server error: certificate has expired.
I_E_SERVER_CERT_IS_INACTIVE = 21557	Server error: certificate is inactive.
I_E_SERVER_CERT_IS_INVALID = 21558	Server error: certificate is invalid.
I_E_SERVER_INVALID_USER_NAME = 21601	Server error: inavlid user name.
I_E_SERVER_NAE_USER_ALREADY_EXISTS = 21602	Server error: NAE user already exists.
I_E_SERVER_MISSING_PASSWD = 21620	Server error: password missing.

I_E_SERVER_INVALID_PASSWORD = 21621	Server error: invalid password.
I_E_SERVER_WEAK_PASSWORD = 21622	Server error: weak password.
I_E_SERVER_MISSING_GROUP_NAME = 21640	Server error: group name missing.
I_E_SERVER_INVALID_GROUP_NAME = 21641	Server error: invalid group name.
I_E_SERVER_GROUP_ALREADY_EXISTS = 21642	Server error: group already exists.
I_E_SERVER_GROUP_DOES_NOT_EXIST = 21643	Server error: group does not exist.
I_E_SERVER_USER_OWNS_KEY = 21660	Server error: user owns the key.
I_E_SERVER_USER_IN_GROUP = 21661	Server error: user in group.
I_E_SERVER_INVALID_USER_OP_LDAP = 21670	Server error: invalid user OP LDAP.
I_E_SERVER_USERMOD_INVALID_EDGESECURE = 21671	Server error: USERMOD INVALID EDGESECURE.
I_E_SERVER_SQL_MISSING_SQL = 21700	Server error: SQL missing.
I_E_SERVER_SQL_MISSING_DATABASE_ALIAS = 21701	Server error: SQL missing database alias.
I_E_SERVER_SQL_BACKEND = 21702	Server error: SQL backend.
I_E_SERVER_SQL_NO_MAPPING = 21703	Server error: no SQL mapping.
I_E_SERVER_BACKEND_COMMUNICATION_FAILURE = 21800	Server error: backend communication failed.
I_E_SERVER_MISSING_MESSAGE_SIZE = 21900	Server error: message size is missing.
I_E_SERVER_INVALID_MESSAGE_SIZE = 21901	Server error: message size is invalid.
I_E_SERVER_MISSING_MESSAGE = 21902	Server error: message is missing.
I_E_SERVER_EXCEEDED_MESSAGE_SIZE = 21903	Server error: message size exceeded.
I_E_ENUM_LENGTH	MUST ALWAYS BE THE LAST ENTRY!!

Description

Error codes run numerically from 0 up, that is, are positive.

Do not use the numeric values in your code! Use the enum.

3.3.2.1 Initialization/Configuration

3.3.3 I_T_InitializationSourceEnum Enumeration

C/C++

```
enum I_T_InitializationSourceEnum {
  I_T_Init_File = 0,
  I_T_Init_Environment = 1
};
```

File

File: cadp_capi.h

Members

Members	Description
I_T_Init_File = 0	Indicates that the initialization call will include the path of the properties file.
I_T_Init_Environment = 1	Indicates that the initialization call will include the name of the environment variable that contains the path of the properties file.

Description

This enum determines how the library is initialized. You can use the CADP_CAPI.properties file, or an environment variable that points to the properties file. Use the I_T_InitializationSource typdef in your code.

3.4 Library Management Opaque Objects

This section describes the Library Management opaque objects available in CADP CAPI KMIP.

Types

Name	Description
I_O_Session	This typedef describes an opaque object representing a session belonging to a single user.

3.4.1 I_O_Session Type

C/C++

```
typedef struct _O_Session * I_O_Session;
```

File

File: cadp_capi.h

Description

This typedef describes an opaque object representing a session belonging to a single user.

3.5 Library Management Macros

This section describes the Library Management macros available in CADP CAPI KMIP.

Macros

Name	Description
FUNCEXP	Export directive
I_T_FALSE	Boolean False value
I_T_TRUE	Boolean True value

3.5.1 FUNCEXP Macro

C/C++

```
#define FUNCEXP
```

File

File: cadp_capi.h

Description

Export directive

3.5.2 I_T_FALSE Macro

C/C++

```
#define I_T_FALSE 0
```

File

File: types.h

Description

Boolean False value

3.5.3 I_T_TRUE Macro

C/C++

```
#define I_T_TRUE 1
```

File

File: types.h

Description

Boolean True value

4 NAE Key Management API

This chapter describes the NAE Key Management API.

4.1 NAE Key Management Functions

This section describes the NAE Key Management functions available in CADP CAPI KMIP. When required, these functions use the NAE XML protocol to communicate with the Key Management Server.

Functions

	Name	Description
	I_C_CreateCustomAttributeList	This function creates an attribute list object.
	I_C_DeleteAttributeList	This function destroys an AttributeList object and release resources.
	I_C_CreateGroupListObject	This function creates a groupList object. A GroupList is a list of user groups and their associated permissions, which allow access to key operations.
	I_C_DeleteGroupListObject	This function deletes an existing groupList object. You must call this function after creating the groupList, or you must pass a null value to avoid a segmentation fault.
	I_C_DeleteKeyInfo	This function deletes a KeyInfo object. You must call this function after creating the KeyInfo, or you must pass a null value to avoid a segmentation fault.
	I_C_GetKeyManagerInfo	This function returns information about the encryption provider. As the client can connect to many servers in active and passive failover, the values returned from this will change randomly for any given call depending on which server connection is used.
	I_C_Random	This function returns random bytes.
	I_C_DeleteStringList	This function deletes a list of strings allocated during call to I_C_FindKey or similar functions.
	I_C_GetNextString	This function iterates through a list of strings returning a current one The function allocates an internal buffer. To prevent memory leaks it must be called in a loop until (ppName == 0). ppName itself should not be deleted.

4.1.1 I_C_CreateCustomAttributeList Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateCustomAttributeList(I_O_AttributeList *
pCustomAttributeList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList * pCustomAttributeList	On output, a pointer to the newly created I_O_AttributeList will be stored in *pAttributeList.

Description

This function creates an attribute list object.

4.1.2 I_C_DeleteAttributeList Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DeleteAttributeList(I_O_AttributeList attributeList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_AttributeList attributeList	The AttributeList object to be destroyed. Both custom and system attribute lists may be passed.

Description

This function destroys an AttributeList object and release resources.

4.1.3 I_C_CreateGroupListObject Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CreateGroupListObject(I_O_GroupList * groupList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_GroupList * groupList	A pointer to the groupList to be created.

Description

This function creates a groupList object.

A GroupList is a list of user groups and their associated permissions, which allow access to key operations.

4.1.4 I_C_DeleteGroupListObject Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DeleteGroupListObject(I_O_GroupList groupList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_GroupList groupList	The groupList object to be deleted.

Description

This function deletes an existing groupList object. You must call this function after creating the groupList, or you must pass a null value to avoid a segmentation fault.

4.1.5 I_C_DeleteKeyInfo Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DeleteKeyInfo(I_O_KeyInfo keyInfo);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_KeyInfo keyInfo	The keyInfo object to be deleted.

Description

This function deletes a KeyInfo object. You must call this function after creating the KeyInfo, or you must pass a null value to avoid a segmentation fault.

4.1.6 I_C_GetKeyManagerInfo Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetKeyManagerInfo(I_O_Session session, I_T_CHAR **
software_version, I_T_CHAR ** library_version, I_T_CHAR ** vendor_ID, I_T_CHAR **
model_number, I_T_CHAR ** serial_number, I_T_CHAR ** datetime);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_T_CHAR ** software_version	[out] The version of the software on the NAE server.
I_T_CHAR ** library_version	[out] Library version.
I_T_CHAR ** vendor_ID	[out] The name of the vendor.
I_T_CHAR ** model_number	[out] The model number of the server (e.g. "i321")
I_T_CHAR ** serial_number	[out] The serial number (or Box ID) of the NAE server.
I_T_CHAR ** datetime	[out] Timestamp from the server in GMT.

Description

This function returns information about the encryption provider.

As the client can connect to many servers in active and passive failover, the values returned from this will change randomly for any given call depending on which server connection is used.

4.1.7 I_C_Random Function

C/C++

```
I_T_RETURN FUNCEXP I_C_Random(I_O_Session session, I_T_UINT randomLength, I_T_BYTE *
outData);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_T_UINT randomLength	The length of the random bytes to be returned.
I_T_BYTE * outData	A buffer to hold the random bytes.

Description

This function returns random bytes.

4.1.8 I_C_DeleteStringList Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DeleteStringList(I_O_StringList pStringList);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
strList	A list of strings which should br destroyed.

Description

This function deletes a list of strings allocated during call to I_C_FindKey or similar functions.

4.1.9 I_C_GetNextString Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetNextString(I_O_StringList strList, void* pIter, void**
pIter_out, I_T_CHAR ** ppName);
```

File

File: cadp_capi_naekeymgmt.h

Parameters

Parameters	Description
I_O_StringList strList	A list of strings.
void* pIter	An input iterator, must be 0 during the first call, in subsequent calls - pass here pIter_out from a previous call.
void** pIter_out	[out] An output iterator, use it as an input iterator for the next call.
I_T_CHAR ** ppName	[out] A current string. 0 indicates the end of list.

Description

This function iterates through a list of strings returning a current one

The function allocates an internal buffer. To prevent memory leaks it must be called in a loop until (ppName == 0). ppName itself should not be deleted.

4.2 NAE Key Management Enumerations

This section describes the NAE Key Management enumerations available in CADP CAPI KMIP.

Enumerations

	Name	Description
	I_T_ExportFormatEnum	You'll use the export format enum to indicate the format of exported certificates. Use the I_T_ExportFormat typedef in your code.
	I_T_KeyParameterTypeEnum	You'll use the key parameter type enum to create new versions of versioned key and modify their states. Use the I_T_KeyParameterType typedef in your code.

	I_T_KeyParameterValueEnum	You'll use the key parameter value enum to create new versions of versioned key and modify their states. Use the I_T_KeyParameterValue typedef in your code.
	I_T_KeyWrapFormatEnum	The Wrap Format defines how the key bytes are encrypted. Use the I_T_KeyWrapFormat typedef in your code.
	I_T_PermissionMaskEnum	The permission mask defines how a key can be employed by a particular group. Multiple masks may be ORed together to define the complete permission granted to a group. **Note:** RSA keys must use I_T_Permission_UsePrivate and I_T_Permission_UsePublic. **Note:** I_T_Permission_Encrypt and I_T_Permission_Decrypt are only valid for symmetric keys; you cannot assign these permissions to an RSA key.

4.2.1 I_T_ExportFormatEnum Enumeration

C/C++

```
enum I_T_ExportFormatEnum {
  I_T_ExportFormat_PEM_PKCS1_CERT_ONLY,
  I_T_ExportFormat_PEM_PKCS1,
  I_T_ExportFormat_PEM_PKCS8,
  I_T_ExportFormat_PKCS12,
  I_T_ExportFormat_PEM_SEC1,
  I_T_ExportFormat_NONE
};
```

File

File: cadp_capi_naekeymgmt.h

Members

Members	Description
I_T_ExportFormat_PEM_PKCS1_CERT_ONLY	The certificate will be exported in PEM format, any private key will be ignored.
I_T_ExportFormat_PEM_PKCS1	The certificate will be exported in PEM format, the private key will be exported in PKCS #1.
I_T_ExportFormat_PEM_PKCS8	The certificate will be exported in PEM format, the private key will be exported in PKCS #8.
I_T_ExportFormat_PKCS12	The certificate and private key will be exported in PKCS#12 format.
I_T_ExportFormat_PEM_SEC1	The certificate and private key will be exported in SEC1 format for EC algorithm.

Description

You'll use the export format enum to indicate the format of exported certificates. Use the I_T_ExportFormat typedef in your code.

4.2.2 I_T_KeyParameterTypeEnum Enumeration

C/C++

```
enum I_T_KeyParameterTypeEnum {
  I_T_KeyLifecycleState = 0,
  I_T_KeyVersion = 1
};
```

File

File: cadp_capi_naekeymgmt.h

Members

Members	Description
I_T_KeyLifecycleState = 0	Change the key state.
I_T_KeyVersion = 1	Create a new version of the key.

Description

You'll use the key parameter type enum to create new versions of versioned key and modify their states. Use the I_T_KeyParameterType typedef in your code.

4.2.3 I_T_KeyParameterValueEnum Enumeration

C/C++

```
enum I_T_KeyParameterValueEnum {
  I_T_KeyParameter_State_Active = 0,
  I_T_KeyParameter_State_Restricted = 10,
  I_T_KeyParameter_State_Retired = 20,
  I_T_KeyParameter_Version_Increment = 100
};
```

File

File: cadp_capi_naekeymgmt.h

Members

Members	Description
I_T_KeyParameter_State_Active = 0	Change the key state to active.
I_T_KeyParameter_State_Restricted = 10	Change the key state to restricted.
I_T_KeyParameter_State_Retired = 20	Change the key state to retired.
I_T_KeyParameter_Version_Increment = 100	Create a new version of the key.

Description

You'll use the key parameter value enum to create new versions of versioned key and modify their states. Use the I_T_KeyParameterValue typedef in your code.

4.2.4 I_T_KeyWrapFormatEnum Enumeration

C/C++

```
enum I_T_KeyWrapFormatEnum {
  I_T_ExportKeyWrapFormat_NONE = 0,
  I_T_ExportKeyWrapFormat_RAW_PKCS1v15 = 1,
  I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA256 = 2,
  I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA384 = 3,
  I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA512 = 4,
  I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA1 = 5
};
```

File

File: cadp_capi_naekeymgmt.h

Members

Members	Description
I_T_ExportKeyWrapFormat_NONE = 0	No Wrap Format is selected.
I_T_ExportKeyWrapFormat_RAW_PKCS1v15 = 1	PKCS1v1.5 encryption will be performed for wrapping the key bytes.
I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA256 = 2	PKCSv2.1 encryption
I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA384 = 3	will be performed
I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA512 = 4	for wrapping key bytes
I_T_ExportKeyWrapFormat_RAW_PKCS1v21_RSAOAEP_SHA1 = 5	PKCSv2.1 encryption with SHA1 used for wrapping key bytes

Description

The Wrap Format defines how the key bytes are encrypted. Use the I_T_KeyWrapFormat typedef in your code.

4.2.5 I_T_PermissionMaskEnum Enumeration

C/C++

```
enum I_T_PermissionMaskEnum {
  I_T_Permission_Encrypt = 0x1,
  I_T_Permission_Decrypt = 0x2,
  I_T_Permission_Sign = 0x4,
  I_T_Permission_SignV = 0x8,
  I_T_Permission_MAC = 0x10,
  I_T_Permission_MACV = 0x20,
  I_T_Permission_UsePrivate = 0x40,
  I_T_Permission_UsePublic = 0x80,
  I_T_Permission_Export = 0x100
};
```

File

File: cadp_capi_naekeymgmt.h

Members

Members	Description
I_T_Permission_Encrypt = 0x1	The group can use the key to encrypt data.
I_T_Permission_Decrypt = 0x2	The group can use the key to decrypt data.
I_T_Permission_Sign = 0x4	The group can use the key to create digital signatures.
I_T_Permission_SignV = 0x8	The group can use the key to verify digital signatures.
I_T_Permission_MAC = 0x10	The group can use the key to create a MAC.
I_T_Permission_MACV = 0x20	The group can use the key to verify a MAC.
I_T_Permission_UsePrivate = 0x40	The group can use the private portion of the RSA key to decrypt data.
I_T_Permission_UsePublic = 0x80	The group can use the public portion of the RSA key to encrypt data.
I_T_Permission_Export = 0x100	The group can export keys from the DataSecure.

Description

The permission mask defines how a key can be employed by a particular group. Multiple masks may be ORed together to define the complete permission granted to a group.

Note: RSA keys must use I_T_Permission_UsePrivate and I_T_Permission_UsePublic.

Note: I_T_Permission_Encrypt and I_T_Permission_Decrypt are only valid for symmetric keys; you cannot assign these permissions to an RSA key.

4.3 NAE Key Management Opaque Objects

This section describes the NAE Key Management opaque objects available in CADP CAPI KMIP.

Types

Name	Description
I_O_AttributeList	An opaque object representing an attribute list
I_O_GroupList	An opaque object representing a list of group permissions for a key.
I_O_KeyInfo	An opaque object representing key information.

4.3.1 I_O_AttributeList Type

C/C++

```
typedef struct _O_AttributeList * I_O_AttributeList;
```

File

File: cadp_capi_naekeymgmt.h

Description

An opaque object representing an attribute list

4.3.2 I_O_GroupList Type

C/C++

```
typedef struct _O_GroupList * I_O_GroupList;
```

File

File: cadp_capi_naekeymgmt.h

Description

An opaque object representing a list of group permissions for a key.

4.3.3 I_O_KeyInfo Type

C/C++

```
typedef struct _O_KeyInfo * I_O_KeyInfo;
```

File

File: cadp_capi_naekeymgmt.h

Description

An opaque object representing key information.

4.4 NAE Key Management Macros

This section describes the NAE Key Management macros available in CADP CAPI KMIP.

Macros

Name	Description
WRAP_FMT_PKCS1v15	Wrap formats for key export WRAP_FMT_[XYZ].
WRAP_FMT_PKCS1v15_LEN	Wrap format string length

4.4.1 WRAP_FMT_PKCS1v15 Macro

C/C++

```
#define WRAP_FMT_PKCS1v15 "PKCS1v1.5"
```

File

File: cadp_capi_naekeymgmt.h

Description

Wrap formats for key export WRAP_FMT_[XYZ].

4.4.2 WRAP_FMT_PKCS1v15_LEN Macro

C/C++

```
#define WRAP_FMT_PKCS1v15_LEN 9
```

File

File: cadp_capi_naekeymgmt.h

Description

Wrap format string length

5 KMIP Key Management API

This chapter describes the KMIP Key Management API. These functions use the KMIP protocol to communicate with the CipherTrust Manager.

5.1 KMIP Key Management Functions

This section describes the KMIP Key Management functions available in CADP CAPI KMIP.

Functions

	Name	Description
	I_KC_ClearAttributeList	This function removes any attributes present in the attribute list.
	I_KC_DeleteAttributeList	This function deletes an Attribute List.
	I_KC_CreateKeyPair	This function creates the RSA Key Pair Object with the KMIP server.
	I_KC_AddAttribute	This function adds an attribute to the Managed Object.
	I_KC_ModifyAttribute	This function modifies an attribute of the Managed Object.
	I_KC_GetAttributesList	This function retrieves the Attribute List of the Managed Object from the KMIP server.
	I_KC_Destroy	This function destroys the specified Managed Object. The Object can be specified by adding Unique Identifier to attributeList.
	I_KC_DeleteAttribute	This function deletes an attribute from the Attribute List.
	I_KC_FreeQueryResponse	This function frees up the memory allocated by the library for holding the Query Response.
	I_KC_FreeManagedObject	This function frees up the memory used by the Managed Object.
	I_KC_FreeUniqueIdentifiers	This function frees up the memory used by I_KS_UniqueIdentifiers.
	I_KC_FreeAttributeNameList	This function frees up the memory used by I_KS_AttributeNameList.
	I_KC_FreeElement	This function frees up the memory used by the element.
	I_KC_CreateBatch	This function creates a batch and returns a pointer to the batch; the batch actually is an API level object, which initially has an empty list of operations. There are flags to specify the behavior of the batch. The order option specifies whether to honor the order.... more
	I_KC_AddOperation	This function allows a user to add one or more operations to the batch. The function generates and returns a unique identifier for each operation in the batch. The arguments are copied and stored in the batch along with the operation.
	I_KC_DeleteOperation	This function searches for the operation as specified by the operation ID and deletes the operation from the batch.

	I_KC_ExecuteBatch	This function triggers the execution of the batch created so far. It has two responsibilities. One is to prepare the request from the batch object as specified in KMIP standards and send the batch to server for execution. The other responsibility is to parse the response received from the server, and prepare the output objects and store them in batch with respect to the operations.
	I_KC_DestroyBatch	This function destroys the batch object and performs cleanup as appropriate.
	I_KC_FreeResultObject	This function destroys the OpOutResult object and performs cleanup as appropriate.
	I_KC_GetResponse	This function searches for the output objects with respect to the operation ID in the batch and returns a pointer to OpOutResult, which holds the result code as well as the output object list.
	I_KC_ResetBatch	This function resets the batch as specified by the flag supporting values defined as follows:... more
	I_KC_Revoke	This is function I_KC_Revoke.
	I_KC_DiscoverVersion	This function gives the KMIP version supported be server.
	I_KC_FreeDiscoverVersionResponse	This function destroys I_KS_SupportedKMIPVersions structure.

5.1.1 I_KC_ClearAttributeList Function

C/C++

```
I_KS_Result FUNCEXP I_KC_ClearAttributeList(I_KO_AttributeList attributeList);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KO_AttributeList attributeList	[in] The handle to the Attribute List Opaque Object to be cleared.

Returns

The result.

Description

This function removes any attributes present in the attribute list.

5.1.2 I_KC_DeleteAttributeList Function

C/C++

```
I_KS_Result FUNCEXP I_KC_DeleteAttributeList(I_KO_AttributeList attributeList);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KO_AttributeList attributeList	[in] The handle to the Attribute List Opaque Object to be deleted.

Returns

The result.

Description

This function deletes an Attribute List.

5.1.3 I_KC_CreateKeyPair Function

C/C++

```
I_KS_Result FUNCEXP I_KC_CreateKeyPair(I_O_Session handle, I_KO_AttributeList
attrListPriv, char* Priv_name, I_KO_AttributeList attrListPub, char * Pub_name);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
char* Priv_name	[in] The Private key name to be created.
char * Pub_name	[in] The Public key name to be created.
attributeListPriv	[in,out] The handle to the Attribute List Opaque Object for Private Key.
attributeListPub	[in,out] The handle to the Attribute List Opaque Object for Public Key.

Returns

The result.

Description

This function creates the RSA Key Pair Object with the KMIP server.

5.1.4 I_KC_AddAttribute Function

C/C++

```
I_KS_Result FUNCEXP I_KC_AddAttribute(I_O_Session handle, I_KO_AttributeList
attributeList);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.

Returns

The result.

Description

This function adds an attribute to the Managed Object.

5.1.5 I_KC_ModifyAttribute Function

C/C++

```
I_KS_Result FUNCEXP I_KC_ModifyAttribute(I_O_Session handle, I_KO_AttributeList
attributeList);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.

Returns

The result.

Description

This function modifies an attribute of the Managed Object.

5.1.6 I_KC_GetAttributesList Function

C/C++

```
I_KS_Result FUNCEXP I_KC_GetAttributesList(I_O_Session handle, I_KO_AttributeList
attributeList, I_KS_AttributeNameList ** attrNameList_pp);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.

I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.
I_KS_AttributeNameList ** attrNameList_pp	[out] A pointer to the I_KS_AttributeNameList structure.

Returns

The result.

Description

This function retrieves the Attribute List of the Managed Object from the KMIP server.

5.1.7 I_KC_Destroy Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Destroy(I_O_Session handle, I_KO_AttributeList
attributeList);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in,out] The handle to the Attribute List Opaque Object.

Returns

The result.

Description

This function destroys the specified Managed Object. The Object can be specified by adding Unique Identifier to attributeList.

5.1.8 I_KC_DeleteAttribute Function

C/C++

```
I_KS_Result FUNCEXP I_KC_DeleteAttribute(I_O_Session handle, I_KO_AttributeList
attributeList, I_T_CHAR * attributes, int index);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
I_KO_AttributeList attributeList	[in] The handle to the Attribute List Opaque Object.

I_T_CHAR * attributes	[in] A pointer to an Attribute Name whose value needs to be deleted from the Key Management Server.
int index	[in] Index of Attribute Names whose values need to be deleted from the Key Management Server.

Description

This function deletes an attribute from the Attribute List.

5.1.9 I_KC_FreeQueryResponse Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeQueryResponse(I_KS_QueryResponse * queryResponse_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KS_QueryResponse * queryResponse_p	[in] A pointer to I_KS_QueryResponse.

Returns

The result.

Description

This function frees up the memory allocated by the library for holding the Query Response.

5.1.10 I_KC_FreeManagedObject Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeManagedObject(I_KS_Object * object_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KS_Object * object_p	[in] A pointer to the I_KS_Object structure allocated by the library.

Returns

The result.

Description

This function frees up the memory used by the Managed Object.

5.1.11 I_KC_FreeUniqueIdentifiers Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeUniqueIdentifiers(I_KS_UniqueIdentifiers*
uniqueIdentifiers_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KS_UniqueIdentifiers* uniqueIdentifiers_p	[in] A pointer to the I_KS_UniqueIdentifiers structure.

Returns

The result.

Description

This function frees up the memory used by I_KS_UniqueIdentifiers.

5.1.12 I_KC_FreeAttributeNameList Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeAttributeNameList(I_KS_AttributeNameList *
attrNameList_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KS_AttributeNameList * attrNameList_p	[in] A pointer to the I_KS_AttributeNameList structure.

Returns

The result.

Description

This function frees up the memory used by I_KS_AttributeNameList.

5.1.13 I_KC_FreeElement Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeElement(I_KS_Element * element);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KS_Element * element	[in] A pointer to the I_KS_Element structure.

Returns

The result.

Description

This function frees up the memory used by the element.

5.1.14 I_KC_CreateBatch Function

C/C++

```
I_KS_Result FUNCEXP I_KC_CreateBatch(BatchHandle * hBatch, I_T_BOOL OrderOption,
BatchErrorOption ErrorConOption);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
BatchHandle * hBatch	[out] A pointer to batch handle.
I_T_BOOL OrderOption	[in] An option to specify whether the order is honored.
BatchErrorOption ErrorConOption	[in] An option to specify to continue when an error occurs.

Returns

The result.

Description

This function creates a batch and returns a pointer to the batch; the batch actually is an API level object, which initially has an empty list of operations. There are flags to specify the behavior of the batch.

The order option specifies whether to honor the order.

- OrderOption TRUE: The server has to execute all operations in the same order in which they are added to the batch.
- OrderOption FALSE: The server is not restricted to follow the order. Therefore, the order of execution may be different.

The error continuation option specifies what to do when an operation in the batch fails.

- I_KT_Batch_Continue 0x00000001: If any operation in the request fails, then server SHALL undo all the previous operations.
- I_KT_Batch_Stop 0x00000002: If an operation fails, then the server SHALL NOT continue processing

subsequent operations in the request. Completed operations SHALL NOT be undone.

- I_KT_Batch_Undo 0x00000003: Return an error for the failed operation, and continue processing subsequent operations in the request.

5.1.15 I_KC_AddOperation Function

C/C++

```
I_KS_Result FUNCEXP I_KC_AddOperation(BatchHandle hBatch, I_KT_OperationCode opcode,
OpArgObject * arg, int argc, I_T_UINT * opid);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
BatchHandle hBatch	[in] The handle to the batch object.
I_KT_OperationCode opcode	[in] The operation code for the operation being added.
OpArgObject * arg	[in] The optional argument list to be passed along with the operation. NULL if no argument is required.
int argc	[in] Number of arguments, that is, count of OpArgObject[].
I_T_UINT * opid	[out] A pointer to hold unique ID (I_T_UINT) for the operation.

Returns

The result.

Description

This function allows a user to add one or more operations to the batch. The function generates and returns a unique identifier for each operation in the batch. The arguments are copied and stored in the batch along with the operation.

5.1.16 I_KC_DeleteOperation Function

C/C++

```
I_KS_Result FUNCEXP I_KC_DeleteOperation(BatchHandle hBatch, I_T_UINT opid);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
BatchHandle hBatch	[in] The handle to the batch object.
I_T_UINT opid	[in] Unique ID for the operation.

Returns

The result.

Description

This function searches for the operation as specified by the operation ID and deletes the operation from the batch.

5.1.17 I_KC_ExecuteBatch Function

C/C++

```
I_KS_Result FUNCEXP I_KC_ExecuteBatch(I_O_Session handle, BatchHandle hBatch);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
BatchHandle hBatch	[in] The handle to the batch object.

Returns

The result.

Description

This function triggers the execution of the batch created so far. It has two responsibilities. One is to prepare the request from the batch object as specified in KMIP standards and send the batch to server for execution. The other responsibility is to parse the response received from the server, and prepare the output objects and store them in batch with respect to the operations.

5.1.18 I_KC_DestroyBatch Function

C/C++

```
I_KS_Result FUNCEXP I_KC_DestroyBatch(BatchHandle hBatch);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
BatchHandle hBatch	[in] The handle to the batch object.

Returns

The result.

Description

This function destroys the batch object and performs cleanup as appropriate.

5.1.19 I_KC_FreeResultObject Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeResultObject(OpOutResult * opresult);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
OpOutResult * opresult	[in] A pointer to the OpOutResult structure.

Returns

The result.

Description

This function destroys the OpOutResult object and performs cleanup as appropriate.

5.1.20 I_KC_GetResponse Function

C/C++

```
I_KS_Result FUNCEXP I_KC_GetResponse(BatchHandle hBatch, I_T_UINT opid, OpOutResult
** opresult_pp);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
BatchHandle hBatch	[in] The handle to the batch object.
I_T_UINT opid	[in] Unique ID for the operation.
OpOutResult ** opresult_pp	[out] A pointer to hold a pointer to the result object for the operation (structure OpOutResult).

Returns

The result.

Description

This function searches for the output objects with respect to the operation ID in the batch and returns a pointer to OpOutResult, which holds the result code as well as the output object list.

5.1.21 I_KC_ResetBatch Function

C/C++

```
I_KS_Result FUNCEXP I_KC_ResetBatch(BatchHandle hBatch, I_T_UINT flag);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
BatchHandle hBatch	[in] The handle to the batch object.
I_T_UINT flag	[in] The flag to determine the reset level.

Returns

The result.

Description

This function resets the batch as specified by the flag supporting values defined as follows:

- I_KT_RESET_OUTPUT_LIST 0x00000001:

For this value of flag, it flushes the output, but keeps the input argument and operation list that can be reused for another execution in the batch.

- I_KT_RESET_ALL_OPERATIONS 0x00000002:

For this value of flag, it deletes the entire list of operations including the operations list, input list, and output list etc. Therefore, the status of the batch is set to empty.

5.1.22 I_KC_Revoke Function

C/C++

```
I_KS_Result FUNCEXP I_KC_Revoke(I_O_Session handle, char const * unique_id, int
code, char const * reason, time_t compromise_occurrence_date);
```

File

File: cadp_capi_kmipkeymgmt.h

Description

This is function I_KC_Revoke.

5.1.23 I_KC_DiscoverVersion Function

C/C++

```
I_KS_Result FUNCEXP I_KC_DiscoverVersion(I_O_Session handle, char *
clientKMIPVersions[], int clientVersionCount, I_KS_SupportedKMIPVersions**
supportedKmipVers_pp);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_O_Session handle	[in] The handle to the session object.
char * clientKMIPVersions[]	[in] KMIP versions supported by client.
int clientVersionCount	[in] Number of KMIP versions supported by client.
I_KS_SupportedKMIPVersions** supportedKmipVers_pp	[out] KMIP versions supported by server.

Returns

The result.

Description

This function gives the KMIP version supported be server.

5.1.24 I_KC_FreeDiscoverVersionResponse Function

C/C++

```
I_KS_Result FUNCEXP I_KC_FreeDiscoverVersionResponse(I_KS_SupportedKMIPVersions *
supportedKmipVers_p);
```

File

File: cadp_capi_kmipkeymgmt.h

Parameters

Parameters	Description
I_KS_SupportedKMIPVersions * supportedKmipVers_p	[in] Structure containing KMIP versions supported by server.

Returns

The result.

Description

This function destroys I_KS_SupportedKMIPVersions structure.

5.2 KMIP Key Management Typedefs

This section describes the KMIP Key Management typedefs available in CADP CAPI KMIP.

Types

Name	Description
I_KT_ENUM	KMIP Enumeration Type.
I_KT_BOOLEAN	KMIP Boolean Type.
I_KT_INTERVAL	KMIP Interval Type.

5.2.1 I_KT_ENUM Type

C/C++

```
typedef I_T_UINT32 I_KT_ENUM;
```

File

File: kmipkeymgmttypes.h

Description

KMIP Enumeration Type.

5.2.2 I_KT_BOOLEAN Type

C/C++

```
typedef I_T_UINT64 I_KT_BOOLEAN;
```

File

File: kmipkeymgmttypes.h

Description

KMIP Boolean Type.

5.2.3 I_KT_INTERVAL Type

C/C++

```
typedef I_T_UINT32 I_KT_INTERVAL;
```

File

File: kmipkeymgmttypes.h

Description

KMIP Interval Type.

5.3 KMIP Key Management Enumerations

This section describes the KMIP Key Management enumerations available in CADP CAPI KMIP.

Enumerations

	Name	Description
	I_KT_AttributeName	Attribute Name Enum.
	I_KT_AttributeValueType	Attribute Value Type Enum.
	I_KT_BlockCipherModeEnum	Block Cipher Mode Enum.
	ArgumentTypeEnum	Argument Type Enum.
	BatchErrorOptionEnum	Batch Error Option Enum.
	BatchStatusEnum	Batch Status Enum.
	I_KT_CryptographicAlgorithmEnum	Cryptographic Algorithm Attribute Enum.
	I_KT_CryptographicUsageMaskEnum	Cryptographic Usage Mask Attribute Enum.
	I_KT_CertificateFormatEnum	Certificate Format Enum.
	I_KT_EncodingOptionEnum	Block Cipher Mode Enum.
	I_KT_HashingAlgorithmEnum	Hashing Algorithms Enum.
	I_KT_KeyCompressionTypeEnum	Key Compression Type Enum.
	I_KT_KeyFormatEnum	Key Format Enum.
	I_KT_LinkType	Link Enum.
	I_KT_NameType	Name Type Enum.
	I_KT_ObjectTypeEnum	Object Type Enum.
	I_KT_OpaqueDataType	Opaque Data Type Enum.
	I_KT_OperationEnum	Operation Type Enum.
	I_KT_OperationCodeEnum	Operation Code Enum.
	I_KT_PaddingEnum	Padding Method Enum.
	I_KT_QueryFunctionEnum	Query Function Type Enum.
	I_KT_ResultReason	Result Reason Enum.
	I_KT_ResultStatus	Result Status Enum.
	I_KT_RevocationReasonCode	Revocation Reason Code Enum.
	I_KT_RoleTypeEnum	Role Type Enum.
	I_KT_SecretData	Secret Data Enum.
	I_KT_State	This is record I_KT_State.
	I_KT_StorageStatusMaskEnum	Storage Status Task Type Enum.

	I_KT_ValueType	KMIP Value Type Enum.
	I_KT_WrappingMethodEnum	Key Wrapping Method Enum.

5.3.1 I_KT_AttributeName Enumeration

C/C++

```
enum I_KT_AttributeName {
  I_KT_AttributeName_UniqueIdentifier = 0x01,
  I_KT_AttributeName_Name = 0x02,
  I_KT_AttributeName_CryptographicAlgorithm = 0x03,
  I_KT_AttributeName_CryptographicLength = 0x04,
  I_KT_AttributeName_ObjectType = 0x05,
  I_KT_AttributeName_CustomAttribute = 0x06,
  I_KT_AttributeName_ContactInformation = 0x07,
  I_KT_AttributeName_Digest = 0x08,
  I_KT_AttributeName_ObjectGroup = 0x09,
  I_KT_AttributeName_InitialDate = 0x0A,
  I_KT_AttributeName_ApplicationSpecificInfo = 0x0B,
  I_KT_AttributeName_Link = 0x0C,
  I_KT_AttributeName_CertificateIdentifier = 0x0D,
  I_KT_AttributeName_CertificateIssuer = 0x0E,
  I_KT_AttributeName_CertificateSubject = 0x0F,
  I_KT_AttributeName_ActivationDate = 0x10,
  I_KT_AttributeName_State = 0x11,
  I_KT_AttributeName_DeactivationDate = 0x12,
  I_KT_AttributeName_ProcessStartDate = 0x13,
  I_KT_AttributeName_ProtectStopDate = 0x14,
  I_KT_AttributeName_DestroyDate = 0x15,
  I_KT_AttributeName_CompromiseOccurrenceDate = 0x16,
  I_KT_AttributeName_CompromiseDate = 0x17,
  I_KT_AttributeName_CryptographicUsageMask = 0x18,
  I_KT_AttributeName_WrappingMethod = 0x19,
  I_KT_AttributeName_BlockCipherMode = 0x1A,
  I_KT_AttributeName_EncodingOption = 0x1B,
  I_KT_AttributeName_Private_UniqueIdentifier = 0x1C,
  I_KT_AttributeName_Public_UniqueIdentifier = 0x1D,
  I_KT_AttributeName_Offset = 0x1E,
  I_KT_AttributeName_IV = 0x1F,
  I_KT_Attribute_Padding = 0x20,
  I_KT_AttributeName_Random_IV = 0x21,
  I_KT_AttributeName_Cipher_Len = 0x22,
  I_KT_AttributeName_IV_Len = 0x23,
  I_KT_AttributeName_Tag_Len = 0x24
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Name Enum.

5.3.2 I_KT_AttributeValueType Enumeration

C/C++

```
enum I_KT_AttributeValueType {
  I_KT_AttributeValueType_TextString = 0x00,
  I_KT_AttributeValueType_Integer = 0x01,
  I_KT_AttributeValueType_Enum = 0x02,
  I_KT_AttributeValueType_Name_S = 0x03,
  I_KT_AttributeValueType_Digest_S = 0x04,
  I_KT_AttributeValueType_DateTime = 0x05,
  I_KT_AttributeValueType_Custom_S = 0x06,
  I_KT_AttributeValueType_ApplicationSpecificInfo_S = 0x07,
  I_KT_AttributeValueType_Link_S = 0x08,
  I_KT_AttributeValueType_CertificateIdentifier_S = 0x09,
  I_KT_AttributeValueType_CertificateIssuer_S = 0x0A,
  I_KT_AttributeValueType_CertificateSubject_S = 0x0B,
  I_KT_AttributeValueType_Bytes = 0x0C
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Value Type Enum.

5.3.3 I_KT_BlockCipherModeEnum Enumeration

C/C++

```
enum I_KT_BlockCipherModeEnum {
  I_KT_Mode_None = 0x00,
  I_KT_Mode_CBC = 0x01,
  I_KT_Mode_ECB = 0x02,
  I_KT_Mode_PCBC = 0x03,
  I_KT_Mode_CFB = 0x04,
  I_KT_Mode_OFB = 0x05,
  I_KT_Mode_CTR = 0x06,
  I_KT_Mode_CMAC = 0x07,
  I_KT_Mode_CCM = 0x08,
  I_KT_Mode_GCM = 0x09,
  I_KT_Mode_CBCMAC = 0x0A,
  I_KT_Mode_XTS = 0x0B,
  I_KT_Mode_AESKeyWrapPadding = 0x0C,
  I_KT_Mode_NISTKeyWrap = 0x0D,
  I_KT_Mode_X9102_AESKW = 0x0E,
  I_KT_Mode_X9102_TDKW = 0x0F,
  I_KT_Mode_X9102_AKW1 = 0x10,
  I_KT_Mode_X9102_AKW2 = 0x11
};
```

File

File: kmipkeymgmttypes.h

Description

Block Cipher Mode Enum.

5.3.4 ArgumentTypeEnum Enumeration

C/C++

```
enum ArgumentTypeEnum {
  I_KT_Agr_AttributeList = 0x00000001,
  I_KT_Agr_Object = 0x00000002,
  I_KT_Agr_ObjectType = 0x00000003,
  I_KT_Agr_GetRequest = 0x00000004,
  I_KT_Agr_StorageStatusMask = 0x00000005,
  I_KT_Agr_QueryFunction = 0x00000006,
  I_KT_Agr_QueryResponse = 0x00000007,
  I_KT_Agr_UnsignedInt = 0x00000008,
  I_KT_Agr_AsciiString = 0x00000009,
  I_KT_Agr_AsciiStringList = 0x0000000A
};
```

File

File: kmipkeymgmttypes.h

Description

Argument Type Enum.

5.3.5 BatchErrorOptionEnum Enumeration

C/C++

```
enum BatchErrorOptionEnum {
  I_KT_Batch_Continue = 0x00000001,
  I_KT_Batch_Stop = 0x00000002,
  I_KT_Batch_Undo = 0x00000003
};
```

File

File: kmipkeymgmttypes.h

Description

Batch Error Option Enum.

5.3.6 BatchStatusEnum Enumeration

C/C++

```
enum BatchStatusEnum {
  I_KT_BatchEmpty = 0x01,
```

```
  I_KT_BatchReady = 0x02,
  I_KT_BatchExecuteSuccess = 0x03,
  I_KT_BatchExecuteFailed = 0x04,
  I_KT_BatchError = 0x05,
  I_KT_FATALERROR = 0xFFFFFFFF
};
```

File

File: kmipkeymgmttypes.h

Description

Batch Status Enum.

5.3.7 I_KT_CryptographicAlgorithmEnum Enumeration

C/C++

```
enum I_KT_CryptographicAlgorithmEnum {
  I_KT_CryptographicAlgorithm_None = 0x00,
  I_KT_CryptographicAlgorithm_DES = 0x01,
  I_KT_CryptographicAlgorithm_3DES = 0x02,
  I_KT_CryptographicAlgorithm_AES = 0x03,
  I_KT_CryptographicAlgorithm_RSA = 0x04,
  I_KT_CryptographicAlgorithm_DSA = 0x05,
  I_KT_CryptographicAlgorithm_ECDSA = 0x06,
  I_KT_CryptographicAlgorithm_HMACSHA1 = 0x07,
  I_KT_CryptographicAlgorithm_HMACSHA224 = 0x08,
  I_KT_CryptographicAlgorithm_HMACSHA256 = 0x09,
  I_KT_CryptographicAlgorithm_HMACSHA384 = 0x0A,
  I_KT_CryptographicAlgorithm_HMACSHA512 = 0x0B,
  I_KT_CryptographicAlgorithm_HMACMD5 = 0x0C,
  I_KT_CryptographicAlgorithm_DH = 0x0D,
  I_KT_CryptographicAlgorithm_ECDH = 0x0E,
  I_KT_CryptographicAlgorithm_ECMQV = 0x0F
};
```

File

File: kmipkeymgmttypes.h

Description

Cryptographic Algorithm Attribute Enum.

5.3.8 I_KT_CryptographicUsageMaskEnum Enumeration

C/C++

```
enum I_KT_CryptographicUsageMaskEnum {
  I_KT_CryptographicUsageMask_None = 0x00000000,
  I_KT_CryptographicUsageMask_Sign = 0x00000001,
  I_KT_CryptographicUsageMask_Verify = 0x00000002,
  I_KT_CryptographicUsageMask_Encrypt = 0x00000004,
  I_KT_CryptographicUsageMask_Decrypt = 0x00000008,
  I_KT_CryptographicUsageMask_WrapKey = 0x00000010,
```

```
  I_KT_CryptographicUsageMask_UnwrapKey = 0x00000020,
  I_KT_CryptographicUsageMask_Export = 0x00000040,
  I_KT_CryptographicUsageMask_MACGenerate = 0x00000080,
  I_KT_CryptographicUsageMask_MACVerify = 0x00000100,
  I_KT_CryptographicUsageMask_DeriveKey = 0x00000200,
  I_KT_CryptographicUsageMask_ContentCommitment = 0x00000400,
  I_KT_CryptographicUsageMask_KeyAgreement = 0x00000800,
  I_KT_CryptographicUsageMask_CertSign = 0x00001000,
  I_KT_CryptographicUsageMask_CRLSign = 0x00002000,
  I_KT_CryptographicUsageMask_GenearteCryptogram = 0x00004000,
  I_KT_CryptographicUsageMask_VerifyCryptogram = 0x00008000,
  I_KT_CryptographicUsageMask_TranslateEncrypt = 0x00010000,
  I_KT_CryptographicUsageMask_TranslateDecrypt = 0x00020000,
  I_KT_CryptographicUsageMask_TranslateWrap = 0x00040000,
  I_KT_CryptographicUsageMask_TranslateUnwrap = 0x00080000
};
```

File

File: kmipkeymgmttypes.h

Description

Cryptographic Usage Mask Attribute Enum.

5.3.9 I_KT_CertificateFormatEnum Enumeration

C/C++

```
enum I_KT_CertificateFormatEnum {
  I_KT_CertificateFormat_X509 = 0x01,
  I_KT_CertificateFormat_PGP = 0x02
};
```

File

File: kmipkeymgmttypes.h

Description

Certificate Format Enum.

5.3.10 I_KT_EncodingOptionEnum Enumeration

C/C++

```
enum I_KT_EncodingOptionEnum {
  I_KT_EncodingOption_No_Encoding = 0x01,
  I_KT_EncodingOption_TTLV_Encoding = 0x02
};
```

File

File: kmipkeymgmttypes.h

Description

Block Cipher Mode Enum.

5.3.11 I_KT_HashingAlgorithmEnum Enumeration

C/C++

```
enum I_KT_HashingAlgorithmEnum {
  I_KT_Hash_None = 0x00,
  I_KT_Hash_MD2 = 0x01,
  I_KT_Hash_MD4 = 0x02,
  I_KT_Hash_MD5 = 0x03,
  I_KT_Hash_SHA1 = 0x04,
  I_KT_Hash_SHA224 = 0x05,
  I_KT_Hash_SHA256 = 0x06,
  I_KT_Hash_SHA384 = 0x07,
  I_KT_Hash_SHA512 = 0x08
};
```

File

File: kmipkeymgmttypes.h

Description

Hashing Algorithms Enum.

5.3.12 I_KT_KeyCompressionTypeEnum Enumeration

C/C++

```
enum I_KT_KeyCompressionTypeEnum {
  I_KT_KeyCompressionType_None = 0x0,
  I_KT_KeyCompressionType_ECPubKey_Uncompressed = 0x1,
  I_KT_KeyCompressionType_ECPubKey_CompressedPrime = 0x2,
  I_KT_KeyCompressionType_ECPubKey_CompressedChar2 = 0x3,
  I_KT_KeyCompressionType_ECPubKey_Hybrid = 0x4
};
```

File

File: kmipkeymgmttypes.h

Description

Key Compression Type Enum.

5.3.13 I_KT_KeyFormatEnum Enumeration

C/C++

```
enum I_KT_KeyFormatEnum {
```

```
  I_KT_KeyFormat_None = 0x00,
  I_KT_KeyFormat_Raw = 0x01,
  I_KT_KeyFormat_Opaque = 0x02,
  I_KT_KeyFormat_PKCS1 = 0x03,
  I_KT_KeyFormat_PKCS8 = 0x04,
  I_KT_KeyFormat_X509 = 0x05,
  I_KT_KeyFormat_ECPrivateKey = 0x06,
  I_KT_KeyFormat_TransparentSymmetricKey = 0x07,
  I_KT_KeyFormat_TransparentDSAPrivateKey = 0x08,
  I_KT_KeyFormat_TransparentDSAPublicKey = 0x09,
  I_KT_KeyFormat_TransparentRSAPrivateKey = 0x0A,
  I_KT_KeyFormat_TransparentRSAPublicKey = 0x0B,
  I_KT_KeyFormat_TransparentDHPrivateKey = 0x0C,
  I_KT_KeyFormat_TransparentDHPublicKey = 0x0D,
  I_KT_KeyFormat_TransparentECDSAPrivateKey = 0x0E,
  I_KT_KeyFormat_TransparentECDSAPublicKey = 0x0F,
  I_KT_KeyFormat_TransparentECDHPrivateKey = 0x10,
  I_KT_KeyFormat_TransparentECDHPublicKey = 0x11,
  I_KT_KeyFormat_TransparentECMQVPrivateKey = 0x12,
  I_KT_KeyFormat_TransparentECMQVPublicKey = 0x13
};
```

File

File: kmipkeymgmttypes.h

Description

Key Format Enum.

5.3.14 I_KT_LinkType Enumeration

C/C++

```
enum I_KT_LinkType {
  I_KT_Certificate_Link = 0x00000101,
  I_KT_PublicKey_Link = 0x00000102,
  I_KT_PrivateKey_Link = 0x00000103,
  I_KT_DerivationBaseObject_Link = 0x00000104,
  I_KT_DerivedKey_Link = 0x00000105,
  I_KT_ReplacementObject_Link = 0x00000106,
  I_KT_ReplacedObject_Link = 0x00000107
};
```

File

File: kmipkeymgmttypes.h

Description

Link Enum.

5.3.15 I_KT_NameType Enumeration

C/C++

```
enum I_KT_NameType {
```

```
  I_KT_NameType_None = 0x00,
  I_KT_NameType_Text = 0x01,
  I_KT_NameType_URI = 0x02
};
```

File

File: kmipkeymgmttypes.h

Description

Name Type Enum.

5.3.16 I_KT_ObjectTypeEnum Enumeration

C/C++

```
enum I_KT_ObjectTypeEnum {
  I_KT_ObjectType_None = 0x00,
  I_KT_ObjectType_Certificate = 0x01,
  I_KT_ObjectType_SymmetricKey = 0x02,
  I_KT_ObjectType_PublicKey = 0x03,
  I_KT_ObjectType_PrivateKey = 0x04,
  I_KT_ObjectType_SplitKey = 0x05,
  I_KT_ObjectType_Template = 0x06,
  I_KT_ObjectType_SecretData = 0x07,
  I_KT_ObjectType_OpaqueObject = 0x08
};
```

File

File: kmipkeymgmttypes.h

Description

Object Type Enum.

5.3.17 I_KT_OpaqueDataType Enumeration

C/C++

```
enum I_KT_OpaqueDataType {
  I_KT_OpaqueDataType_None = 0x00
};
```

File

File: kmipkeymgmttypes.h

Description

Opaque Data Type Enum.

5.3.18 I_KT_OperationEnum Enumeration

C/C++

```
enum I_KT_OperationEnum {
  I_KT_Operation_None = 0x00000000,
  I_KT_Operation_Create = 0x00000001,
  I_KT_Operation_CreateKeyPair = 0x00000002,
  I_KT_Operation_Register = 0x00000003,
  I_KT_Operation_ReKey = 0x00000004,
  I_KT_Operation_DeriveKey = 0x00000005,
  I_KT_Operation_Certify = 0x00000006,
  I_KT_Operation_ReCertify = 0x00000007,
  I_KT_Operation_Locate = 0x00000008,
  I_KT_Operation_Check = 0x00000009,
  I_KT_Operation_Get = 0x0000000a,
  I_KT_Operation_GetAttributes = 0x0000000b,
  I_KT_Operation_GetAttributesList = 0x0000000c,
  I_KT_Operation_AddAttribute = 0x0000000d,
  I_KT_Operation_ModifyAttribute = 0x0000000e,
  I_KT_Operation_DeleteAttribute = 0x0000000f,
  I_KT_Operation_ObtainLease = 0x00000010,
  I_KT_Operation_GetUsageAllocation = 0x00000011,
  I_KT_Operation_Activate = 0x00000012,
  I_KT_Operation_Revoke = 0x00000013,
  I_KT_Operation_Destroy = 0x00000014,
  I_KT_Operation_Archive = 0x00000015,
  I_KT_Operation_Recover = 0x00000016,
  I_KT_Operation_Validate = 0x00000017,
  I_KT_Operation_Query = 0x00000018,
  I_KT_Operation_Cancel = 0x00000019,
  I_KT_Operation_Poll = 0x0000001a,
  I_KT_Operation_Notify = 0x0000001b,
  I_KT_Operation_Put = 0x0000001c,
  I_KT_Operation_ReKey_KeyPair = 0x0000001d,
  I_KT_Operation_DiscoverVersions = 0x0000001e
};
```

File

File: kmipkeymgmttypes.h

Description

Operation Type Enum.

5.3.19 I_KT_OperationCodeEnum Enumeration

C/C++

```
enum I_KT_OperationCodeEnum {
  I_KT_Operation_Code_None = 0x00000000,
  I_KT_Operation_Code_Create = 0x00000001,
  I_KT_Operation_Code_Register = 0x00000003,
  I_KT_Operation_Code_Locate = 0x00000008,
  I_KT_Operation_Code_Get = 0x0000000a,
```

```
  I_KT_Operation_Code_GetAttributes = 0x0000000b,
  I_KT_Operation_Code_GetAttributesList = 0x0000000c,
  I_KT_Operation_Code_AddAttribute = 0x0000000d,
  I_KT_Operation_Code_ModifyAttribute = 0x0000000e,
  I_KT_Operation_Code_DeleteAttribute = 0x0000000f,
  I_KT_Operation_Code_Destroy = 0x00000014,
  I_KT_Operation_Code_Query = 0x00000018
};
```

File

File: kmipkeymgmttypes.h

Description

Operation Code Enum.

5.3.20 I_KT_PaddingEnum Enumeration

C/C++

```
enum I_KT_PaddingEnum {
  I_KT_Padding_None = 0x01,
  I_KT_Padding_OAEP = 0x02,
  I_KT_Padding_PKCS5 = 0x03,
  I_KT_Padding_SSL3 = 0x04,
  I_KT_Padding_Zeros = 0x05,
  I_KT_Padding_ANSI_X923 = 0x06,
  I_KT_Padding_ISO_10126 = 0x07,
  I_KT_Padding_PKCS1_v15 = 0x08,
  I_KT_Padding_X931 = 0x09,
  I_KT_Padding_PSS = 0x0A
};
```

File

File: kmipkeymgmttypes.h

Description

Padding Method Enum.

5.3.21 I_KT_QueryFunctionEnum Enumeration

C/C++

```
enum I_KT_QueryFunctionEnum {
  I_KT_QueryFunction_None = 0x0000,
  I_KT_QueryFunction_Operations = 0x0001,
  I_KT_QueryFunction_Objects = 0x0002,
  I_KT_QueryFunction_ServerInformation = 0x0003,
  I_KT_QueryFunction_ApplicationNameSpaces = 0x0004,
  I_KT_QueryFunction_ExtensionList = 0x0005,
  I_KT_QueryFunction_ExtensionMap = 0x0006
};
```

File

File: kmipkeymgmttypes.h

Description

Query Function Type Enum.

5.3.22 I_KT_ResultReason Enumeration

C/C++

```
enum I_KT_ResultReason {
  I_KT_ResultReason_NoError = 0x00,
  I_KT_ResultReason_ItemNotFound = 0x01,
  I_KT_ResultReason_ResponseTooLarge = 0x02,
  I_KT_ResultReason_AuthenticationNotSuccessful = 0x03,
  I_KT_ResultReason_InvalidMessage = 0x04,
  I_KT_ResultReason_OperationNotSupported = 0x05,
  I_KT_ResultReason_MissingData = 0x06,
  I_KT_ResultReason_InvalidField = 0x07,
  I_KT_ResultReason_FeatureNotSupported = 0x08,
  I_KT_ResultReason_OperationCanceledByRequester = 0x09,
  I_KT_ResultReason_CryptographicFailure = 0x0A,
  I_KT_ResultReason_IllegalOperation = 0x0B,
  I_KT_ResultReason_PermissionDenied = 0x0C,
  I_KT_ResultReason_ObjectArchived = 0x0D,
  I_KT_ResultReason_IndexOutOfBounds = 0x0E,
  I_KT_ResultReason_ApplicationNamespaceNotSupported = 0x0F,
  I_KT_ResultReason_KeyFormatTypeNotSupported = 0x10,
  I_KT_ResultReason_KeyCompressionTypeNotSupported = 0x11,
  I_KT_ResultReason_EncodingOptionError = 0x12,
  I_KT_ResultReason_GeneralFailure = 0x100,
  I_KT_ResultReason_InvalidSessionHandle = 0x80000000,
  I_KT_ResultReason_InvalidParameter = 0x80000001,
  I_KT_ResultReason_InvalidConfiguration = 0x80000002,
  I_KT_ResultReason_InternalError = 0x80000003,
  I_KT_ResultReason_MissingMandatoryAttribute = 0x80000004,
  I_KT_ResultReason_OperationNotSupprted = 0x80000005,
  I_KT_ResultReason_InvalidResponse = 0x80000006,
  I_KT_ResultReason_InvalidAttributeValue = 0x80000007,
  I_KT_ResultReason_OutOfMemory = 0x80000008,
  I_KT_ResultReason_MultiInstancesNotPermitted = 0x80000009,
  I_KT_ResultReason_AttributeValueAtIndexExists = 0x8000000a,
  I_KT_ResultReason_AttributeValueNotFound = 0x8000000b,
  I_KT_ResultReason_SerializationError = 0x8000000c,
  I_KT_ResultReason_ConnectionWriteError = 0x81000000,
  I_KT_ResultReason_ConnectionReadError = 0x81000001,
  I_KT_ResultReason_ConnectionConnectError = 0x81000002
};
```

File

File: kmiperr.h

Members

Members	Description
I_KT_ResultReason_NoError = 0x00	No Error.
I_KT_ResultReason_ItemNotFound = 0x01	Item Not Found.
I_KT_ResultReason_ResponseTooLarge = 0x02	Response Too Large.
I_KT_ResultReason_AuthenticationNotSuccessful = 0x03	Authentication Not Successful.
I_KT_ResultReason_InvalidMessage = 0x04	Invalid Message.
I_KT_ResultReason_OperationNotSupported = 0x05	Operation Not Supported.
I_KT_ResultReason_MissingData = 0x06	Missing Data.
I_KT_ResultReason_InvalidField = 0x07	Invalid Field.
I_KT_ResultReason_FeatureNotSupported = 0x08	Feature Not Supported.
I_KT_ResultReason_OperationCanceledByRequester = 0x09	Operation Canceled By Requester.
I_KT_ResultReason_CryptographicFailure = 0x0A	Cryptographic Failure.
I_KT_ResultReason_IllegalOperation = 0x0B	Illegal Operation.
I_KT_ResultReason_PermissionDenied = 0x0C	Permission Denied.
I_KT_ResultReason_ObjectArchived = 0x0D	Object Archived.
I_KT_ResultReason_IndexOutOfBounds = 0x0E	Index Out Of Bounds.
I_KT_ResultReason_ApplicationNamespaceNotSupported = 0x0F	Application Namespace Not Supported.
I_KT_ResultReason_KeyFormatTypeNotSupported = 0x10	Key Format Type Not Supported.
I_KT_ResultReason_KeyCompressionTypeNotSupported = 0x11	Key Compression Type Not Supported.
I_KT_ResultReason_EncodingOptionError = 0x12	Encoding Option Error
I_KT_ResultReason_GeneralFailure = 0x100	General Failure.
I_KT_ResultReason_InvalidSessionHandle = 0x80000000	Invalid Session Handle.
I_KT_ResultReason_InvalidParameter = 0x80000001	Inavlid Parameter.
I_KT_ResultReason_InvalidConfiguration = 0x80000002	Invalid Configuration.
I_KT_ResultReason_InternalError = 0x80000003	Internal Error.
I_KT_ResultReason_MissingMandatoryAttribute = 0x80000004	Missing Mandatory Attribute.
I_KT_ResultReason_OperationNotSupprted = 0x80000005	Operation Not Supported.
I_KT_ResultReason_InvalidResponse = 0x80000006	Invalid Response.
I_KT_ResultReason_InvalidAttributeValue = 0x80000007	Invalid Attribute Value.
I_KT_ResultReason_OutOfMemory = 0x80000008	Out Of Memory.
I_KT_ResultReason_MultiInstancesNotPermitted = 0x80000009	Multiple Instances Not Permitted.
I_KT_ResultReason_AttributeValueAtIndexExists = 0x8000000a	Attribute Value At Index Exists.
I_KT_ResultReason_AttributeValueNotFound = 0x8000000b	Attribute Value Not Found.
I_KT_ResultReason_SerializationError = 0x8000000c	Serialization Error.
I_KT_ResultReason_ConnectionWriteError = 0x81000000	Connection Write Error.

I_KT_ResultReason_ConnectionReadError = 0x81000001	Connection Read Error.
I_KT_ResultReason_ConnectionConnectError = 0x81000002	Connection Connect Error.

Description

Result Reason Enum.

5.3.23 I_KT_ResultStatus Enumeration

C/C++

```
enum I_KT_ResultStatus {
  I_KT_ResultStatus_Success = 0x00,
  I_KT_ResultStatus_OperationFailed = 0x01,
  I_KT_ResultStatus_OperationPending = 0x02,
  I_KT_ResultStatus_OperationUndone = 0x03,
  I_KT_ResultStatus_Library_Error = 0x80000000,
  I_KT_ResultStatus_AttribNotFound = 0x80000001
};
```

File

File: kmiperr.h

Members

Members	Description
I_KT_ResultStatus_Success = 0x00	Success.
I_KT_ResultStatus_OperationFailed = 0x01	Operation Failed.
I_KT_ResultStatus_OperationPending = 0x02	Operation Pending.
I_KT_ResultStatus_OperationUndone = 0x03	Operation Undone.
I_KT_ResultStatus_Library_Error = 0x80000000	Library Error.
I_KT_ResultStatus_AttribNotFound = 0x80000001	Attribute retrieval failed, can be ignored as error

Description

Result Status Enum.

5.3.24 I_KT_RevocationReasonCode Enumeration

C/C++

```
enum I_KT_RevocationReasonCode {
  I_KT_RevocationReasonCode_None = 0x00,
  I_KT_RevocationReasonCode_Unspecified = 0x01,
  I_KT_RevocationReasonCode_KeyCompromise = 0x02,
  I_KT_RevocationReasonCode_CACompromise = 0x03,
  I_KT_RevocationReasonCode_AffiliationChanged = 0x04,
  I_KT_RevocationReasonCode_Superseded = 0x05,
  I_KT_RevocationReasonCode_CessationOperation = 0x06,
  I_KT_RevocationReasonCode_PrivilegeWithdrawn = 0x07
};
```

File

File: kmipkeymgmttypes.h

Description

Revocation Reason Code Enum.

5.3.25 I_KT_RoleTypeEnum Enumeration

C/C++

```
enum I_KT_RoleTypeEnum {
  I_KT_RoleType_None = 0x00,
  I_KT_RoleType_BDK = 0x01,
  I_KT_RoleType_CVK = 0x02,
  I_KT_RolTypee_DEK = 0x03,
  I_KT_RoleType_MKAC = 0x04,
  I_KT_RoleType_MKSMC = 0x05,
  I_KT_RoleType_MKSMI = 0x06,
  I_KT_RoleType_MKDAC = 0x07,
  I_KT_RoleType_MKDN = 0x08,
  I_KT_RoleType_MKCP = 0x09,
  I_KT_RoleType_MKOTH = 0x0A,
  I_KT_RoleType_KEK = 0x0B,
  I_KT_RoleType_MAC16609 = 0x0C,
  I_KT_RoleType_MAC97971 = 0x0D,
  I_KT_RoleType_MAC97972 = 0x0E,
  I_KT_RoleType_MAC97973 = 0x0F,
  I_KT_RoleType_MAC97974 = 0x10,
  I_KT_RoleType_MAC97975 = 0x11,
  I_KT_RoleType_ZPK = 0x12,
  I_KT_RoleType_PVKIBM = 0x13,
  I_KT_RoleType_PVKPVV = 0x14,
  I_KT_RoleType_PVKOTH = 0x15
};
```

File

File: kmipkeymgmttypes.h

Description

Role Type Enum.

5.3.26 I_KT_SecretData Enumeration

C/C++

```
enum I_KT_SecretData {
  I_KT_SecretData_None = 0x00,
  I_KT_SecretData_Password = 0x01,
  I_KT_SecretData_Seed = 0x02
};
```

File

File: kmipkeymgmttypes.h

Description

Secret Data Enum.

5.3.27 I_KT_State Enumeration

C/C++

```
enum I_KT_State {
  I_KT_State_PreActive = 0x01,
  I_KT_State_Active = 0x02,
  I_KT_State_Deactivated = 0x03,
  I_KT_State_Compromised = 0x04,
  I_KT_State_Destroyed = 0x05,
  I_KT_State_Destroyed_Compromised = 0x06
};
```

File

File: kmipkeymgmttypes.h

Description

This is record I_KT_State.

5.3.28 I_KT_StorageStatusMaskEnum Enumeration

C/C++

```
enum I_KT_StorageStatusMaskEnum {
  I_KT_StorageStatus_None = 0x00,
  I_KT_StorageStatus_Online = 0x01,
  I_KT_StorageStatus_Archival = 0x02
};
```

File

File: kmipkeymgmttypes.h

Description

Storage Status Task Type Enum.

5.3.29 I_KT_ValueType Enumeration

C/C++

```
enum I_KT_ValueType {
  I_KT_Invalid,
```

```
    I_KT_Struct,
    I_KT_Integer,
    I_KT_LongInteger,
    I_KT_Enumeration,
    I_KT_Boolean,
    I_KT_TextString,
    I_KT_ByteString,
    I_KT_DateTime,
    I_KT_Interval,
    I_KT_BigInteger
};
```

File

File: kmipkeymgmttypes.h

Description

KMIP Value Type Enum.

5.3.30 I_KT_WrappingMethodEnum Enumeration

C/C++

```
enum I_KT_WrappingMethodEnum {
    I_KT_WrappingMethod_Encrypt = 0x01,
    I_KT_WrappingMethod_Mac_Sign = 0x02,
    I_KT_WrappingMethod_Enc_Mac_Sign = 0x03,
    I_KT_WrappingMethod_Mac_Sign_Enc = 0x04,
    I_KT_WrappingMethod_Tr_31 = 0x05
};
```

File

File: kmipkeymgmttypes.h

Description

Key Wrapping Method Enum.

5.4 KMIP Key Management Structs

This section describes the KMIP Key Management structs available in CADP CAPI KMIP.

Structures

	Name	Description
	I_KS_Attribute	Attribute Structure.
	I_KS_AttributeDigest	Attribute Digest Structure.
	I_KS_AttributeName	Attribute Name Structure.
	I_KS_Link	Attribute Link Structure.
	I_KS_ApplicationSpecificInformation	Attribute Application Specific Information Structure.

	I_KS_AttributeNameList	Structure containing array of Attribute Names.
	I_KS_AttributeValue	Attribute Value Structure.
	I_KS_ByteString	KMIP Structs... more
	I_KS_ElementStruct	KMIP TTLV Element structure.
	I_KS_GetRequest	Structure containing parameters for Get Request.
	I_KS_KeyBlock	Key Block Base Object.
	I_KS_Object	KMIP Managed Object Strucutre.
	I_KS_QueryResponse	Query Response Strucutre.
	I_KS_SStruct	KMIP Struct Type strucutre for storing structure.
	I_KS_UniqueIdentifiers	Structure containing array of Unique Identifiers.
	I_KS_ValueStruct	KMIP TTLV Value.
	I_KS_Result	Result Struct.
	_I_KS_AsciiString	ASCII String Structure.
	_OpArgObject	Operation Argument Object Structure.
	_OpOutResult	Operation Out Result Structure.
	I_KS_AsciiString	ASCII String Structure.
	OpArgObject	Operation Argument Object Structure.
	OpOutObject	Operation Argument Object Structure.
	OpOutResult	Operation Out Result Structure.
	I_KS_CertificateIdentifier	Attribute Certificate Identifier Structure.
	I_KS_CertificateIssuer	Attribute Certificate Issuer Structure.
	I_KS_CertificateSubject	Attribute Certificate Subject Structure.
	I_KS_SupportedKMIPVersions	Structure containing array of Supported KMIP Versions.

Types

Name	Description
I_KS_Element	This is type I_KS_Element.
I_KS_Struct	This is type I_KS_Struct.
I_KS_Value	This is type I_KS_Value.

5.4.1 I_KS_Attribute Structure

C/C++

```
struct I_KS_Attribute {
  I_T_CHAR * attributeName;
  I_KS_AttributeValue attributeValue;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Structure.

5.4.2 I_KS_AttributeDigest Structure

C/C++

```
struct I_KS_AttributeDigest {
  I_KS_ByteString byteString_s;
  I_KT_HashingAlgorithm algo_t;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Digest Structure.

5.4.3 I_KS_AttributeName Structure

C/C++

```
struct I_KS_AttributeName {
  I_T_CHAR * name_p;
  I_KT_NameType nameType_t;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Name Structure.

5.4.4 I_KS_Link Structure

C/C++

```
struct I_KS_Link {
  I_KT_LinkType linktype_t;
  I_T_CHAR * uniqueIdentifiers_p;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Link Structure.

5.4.5 I_KS_ApplicationSpecificInformation Structure

C/C++

```
struct I_KS_ApplicationSpecificInformation {
  I_T_CHAR * namespace_p;
  I_T_CHAR * data_p;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Application Specific Information Structure.

5.4.6 I_KS_AttributeNameList Structure

C/C++

```
struct I_KS_AttributeNameList {
  I_T_CHAR ** attrlistname_pp;
  I_T_UINT count;
};
```

File

File: kmipkeymgmttypes.h

Description

Structure containing array of Attribute Names.

5.4.7 I_KS_AttributeValue Structure

C/C++

```
struct I_KS_AttributeValue {
  I_KT_AttributeValueType valueType_t;
  I_T_INT32 index;
  union {
      I_KS_ByteString bytes;
      I_T_CHAR * textStringVal_p;
      I_KT_ENUM enumVal;
      I_T_INT32 integerVal;
      I_KS_AttributeName name_s;
      I_KS_AttributeDigest digest_s;
      time_t dateTimeVal;
      I_KS_Value custom_s;
      I_KS_ApplicationSpecificInformation asi_s;
      I_KS_Link link_s;
      I_KS_CertificateIdentifier cert_id_s;
      I_KS_CertificateIssuer cert_iss_s;
```

```
        I_KS_CertificateSubject cert_subj_s;
    } value_u;
};
```

File

File: kmipkeymgmttypes.h

Members

Members	Description
I_T_INT32 index;	Index
I_KS_ByteString bytes;	I_KT_AttributeValueType_Bytes
I_T_CHAR * textStringVal_p;	I_KT_AttributeValueType_TextString
I_KT_ENUM enumVal;	I_KT_AttributeValueType_Enum
I_T_INT32 integerVal;	I_KT_AttributeValueType_Integer
I_KS_AttributeName name_s;	I_KT_AttributeValueType_Name_S
I_KS_AttributeDigest digest_s;	I_KT_AttributeValueType_Digest_S
time_t dateTimeVal;	I_KT_AttributeValueType_DateTime
I_KS_Value custom_s;	I_KT_AttributeValueType_Custom_S
I_KS_ApplicationSpecificInformation asi_s;	I_KT_AttributeValueType_ApplicationSpecificInfo_S
I_KS_Link link_s;	I_KT_AttributeValueType_Link_S
I_KS_CertificateIdentifier cert_id_s;	I_KT_AttributeName_CertificateIdentifier_S
I_KS_CertificateIssuer cert_iss_s;	I_KT_AttributeName_CertificateIssuer_S

Description

Attribute Value Structure.

5.4.8 I_KS_ByteString Structure

C/C++

```
struct I_KS_ByteString {
  I_T_BYTE * byteString_p;
  I_T_UINT byteStringLen;
};
```

File

File: kmipkeymgmttypes.h

Description

KMIP Structs

****

- KMIP Byte String Type structure for storing Byte String.

5.4.9 I_KS_ElementStruct Structure

C/C++

```
struct I_KS_ElementStruct {
  I_T_UINT32 tag;
  struct I_KS_ValueStruct value_s;
};
```

File

File: kmipkeymgmttypes.h

Description

KMIP TTLV Element structure.

5.4.10 I_KS_GetRequest Structure

C/C++

```
struct I_KS_GetRequest {
  I_KT_KeyFormat keyFormat_t;
};
```

File

File: kmipkeymgmttypes.h

Description

Structure containing parameters for Get Request.

5.4.11 I_KS_KeyBlock Structure

C/C++

```
struct I_KS_KeyBlock {
  I_KT_KeyFormat keyFormat_t;
  I_KT_KeyCompressionType keyCompress_t;
  I_KT_CryptographicAlgorithm keyAlgo_t;
  I_T_UINT keyLength;
  I_KS_ByteString keyBytes_s;
};
```

File

File: kmipkeymgmttypes.h

Description

Key Block Base Object.

5.4.12 I_KS_Object Structure

C/C++

```
struct I_KS_Object {
  I_KT_ObjectType objectType_t;
  union {
      struct symmetricKey {
            I_KS_KeyBlock keyBlock_s;
          } symmetricKey_s;
      struct secretData {
            I_KT_SecretData secretDataType;
            I_KS_KeyBlock keyBlock_s;
          } secretData_s;
      struct privateKey {
            I_KS_KeyBlock keyBlock_s;
          } privateKey_s;
      struct publicKey {
            I_KS_KeyBlock keyBlock_s;
          } publicKey_s;
      struct certificate {
            I_KT_CertificateFormat certFormat;
            I_KS_ByteString keyBytes_s;
          } certificate_s;
    } object_u;
};
```

File

File: kmipkeymgmttypes.h

Description

KMIP Managed Object Strucutre.

5.4.13 I_KS_QueryResponse Structure

C/C++

```
struct I_KS_QueryResponse {
  I_KT_Operation * operation_p;
  I_T_UINT operationsCount;
  I_KT_ObjectType * objectType_p;
  I_T_UINT objectTypeCount;
  I_T_CHAR * vendorIdentification_p;
  I_KS_Element * serverInformationElements_p;
  I_T_CHAR ** applicationNamespace_pp;
  I_T_UINT applicationNamespaceCount;
  I_KS_Struct * extensionList_p;
  I_T_UINT extensionListCount;
  I_T_CHAR ** extensionMap_pp;
  I_T_UINT extensionMapCount;
};
```

File

File: kmipkeymgmttypes.h

Description

Query Response Strucutre.

5.4.14 I_KS_SStruct Structure

C/C++

```
struct I_KS_SStruct {
  struct I_KS_ElementStruct ** fields;
  I_T_UINT nFields;
};
```

File

File: kmipkeymgmttypes.h

Description

KMIP Struct Type strucutre for storing structure.

5.4.15 I_KS_UniqueIdentifiers Structure

C/C++

```
struct I_KS_UniqueIdentifiers {
  I_T_CHAR ** uniqueIdentifiers_pp;
  I_T_UINT count;
};
```

File

File: kmipkeymgmttypes.h

Description

Structure containing array of Unique Identifiers.

5.4.16 I_KS_ValueStruct Structure

C/C++

```
struct I_KS_ValueStruct {
  I_KT_ValueType type;
  union value {
      struct I_KS_SStruct* structVal;
      I_T_INT32 integerVal;
      I_T_INT64 longIntegerVal;
      I_KT_ENUM enumVal;
```

```
        I_KT_BOOLEAN boolVal;
        I_T_CHAR * textStringVal_p;
        I_KS_ByteString byteString_s;
        time_t dateTimeVal;
        I_KT_INTERVAL intervalVal;
        I_KS_ByteString bigIntegerVal_s;
    } value_u;
};
```

File

File: kmipkeymgmttypes.h

Members

Members	Description
union value { struct I_KS_SStruct* structVal; I_T_INT32 integerVal; I_T_INT64 longIntegerVal; I_KT_ENUM enumVal; I_KT_BOOLEAN boolVal; I_T_CHAR * textStringVal_p; I_KS_ByteString byteString_s; time_t dateTimeVal; I_KT_INTERVAL intervalVal; I_KS_ByteString bigIntegerVal_s; } value_u;	Value Union
struct I_KS_SStruct* structVal;	I_KT_Struct
I_T_INT32 integerVal;	I_KT_Integer
I_T_INT64 longIntegerVal;	I_KT_LongInteger
I_KT_ENUM enumVal;	I_KT_Enumeration
I_KT_BOOLEAN boolVal;	I_KT_Boolean
I_T_CHAR * textStringVal_p;	I_KT_TextString
I_KS_ByteString byteString_s;	I_KT_ByteString
time_t dateTimeVal;	I_KT_DateTime
I_KT_INTERVAL intervalVal;	I_KT_Interval
I_KS_ByteString bigIntegerVal_s;	I_KT_BigInteger

Description

KMIP TTLV Value.

5.4.17 I_KS_Element Type

C/C++

```
typedef struct I_KS_ElementStruct I_KS_Element;
```

File

File: kmipkeymgmttypes.h

Description

This is type I_KS_Element.

5.4.18 I_KS_Struct Type

C/C++

```
typedef struct I_KS_SStruct I_KS_Struct;
```

File

File: kmipkeymgmttypes.h

Description

This is type I_KS_Struct.

5.4.19 I_KS_Value Type

C/C++

```
typedef struct I_KS_ValueStruct I_KS_Value;
```

File

File: kmipkeymgmttypes.h

Description

This is type I_KS_Value.

5.4.20 I_KS_Result Structure

C/C++

```
typedef struct {
  I_KT_ResultStatus status;
  I_KT_ResultReason reason;
} I_KS_Result;
```

File

File: kmiperr.h

Members

Members	Description
I_KT_ResultStatus status;	Result Status.
I_KT_ResultReason reason;	Result Reason.

Description

Result Struct.

5.4.21 _I_KS_AsciiString Structure

C/C++

```
struct _I_KS_AsciiString {
  I_T_CHAR * pbStr;
  I_T_UINT size;
};
```

File

File: kmipkeymgmttypes.h

Description

ASCII String Structure.

5.4.22 _OpArgObject Structure

C/C++

```
struct _OpArgObject {
  ArgumentType type;
  struct {
      I_KO_AttributeList attributeList;
      I_KS_QueryResponse * queryResponse_p;
      I_KS_AsciiString string;
      I_KT_ObjectType objectType;
      struct {
            I_KS_Object * object_p;
            I_T_UINT object_c;
          };
      I_KS_GetRequest getRequest_p;
      struct {
            I_KT_QueryFunction * queryFunctions_p;
            I_T_UINT queryFunctions_c;
          };
      I_KT_StorageStatusMask storageMask;
      struct {
            I_KS_AsciiString * AsciiStringList_p;
            I_T_UINT AsciiStringList_c;
          };
      I_T_UINT UInterger;
    };
};
```

File

File: kmipkeymgmttypes.h

Description

Operation Argument Object Structure.

5.4.23 _OpOutResult Structure

C/C++

```
struct _OpOutResult {
  I_KS_Result opresult;
  I_T_UINT outobjcount;
  OpOutObject * outobj;
};
```

File

File: kmipkeymgmttypes.h

Description

Operation Out Result Structure.

5.4.24 I_KS_AsciiString Structure

C/C++

```
typedef struct _I_KS_AsciiString {
  I_T_CHAR * pbStr;
  I_T_UINT size;
} I_KS_AsciiString;
```

File

File: kmipkeymgmttypes.h

Description

ASCII String Structure.

5.4.25 OpArgObject Structure

C/C++

```
typedef struct _OpArgObject {
  ArgumentType type;
  struct {
      I_KO_AttributeList attributeList;
      I_KS_QueryResponse * queryResponse_p;
      I_KS_AsciiString string;
      I_KT_ObjectType objectType;
      struct {
            I_KS_Object * object_p;
            I_T_UINT object_c;
```

```
            }
        I_KS_GetRequest getRequest_p;
        struct {
              I_KT_QueryFunction * queryFunctions_p;
              I_T_UINT queryFunctions_c;
            }
        I_KT_StorageStatusMask storageMask;
        struct {
              I_KS_AsciiString * AsciiStringList_p;
              I_T_UINT AsciiStringList_c;
            }
        I_T_UINT UInterger;
      }
} OpArgObject, OpOutObject;
```

File

File: kmipkeymgmttypes.h

Description

Operation Argument Object Structure.

5.4.26 OpOutObject Structure

C/C++

```
typedef struct _OpArgObject {
  ArgumentType type;
  struct {
      I_KO_AttributeList attributeList;
      I_KS_QueryResponse * queryResponse_p;
      I_KS_AsciiString string;
      I_KT_ObjectType objectType;
      struct {
            I_KS_Object * object_p;
            I_T_UINT object_c;
          }
      I_KS_GetRequest getRequest_p;
      struct {
            I_KT_QueryFunction * queryFunctions_p;
            I_T_UINT queryFunctions_c;
          }
      I_KT_StorageStatusMask storageMask;
      struct {
            I_KS_AsciiString * AsciiStringList_p;
            I_T_UINT AsciiStringList_c;
          }
      I_T_UINT UInterger;
    }
} OpArgObject, OpOutObject;
```

File

File: kmipkeymgmttypes.h

Description

Operation Argument Object Structure.

5.4.27 OpOutResult Structure

C/C++

```
typedef struct _OpOutResult {
  I_KS_Result opresult;
  I_T_UINT outobjcount;
  OpOutObject * outobj;
} OpOutResult;
```

File

File: kmipkeymgmttypes.h

Description

Operation Out Result Structure.

5.4.28 I_KS_CertificateIdentifier Structure

C/C++

```
struct I_KS_CertificateIdentifier {
  I_T_CHAR * issuer_p;
  I_T_CHAR * sernum_p;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Certificate Identifier Structure.

5.4.29 I_KS_CertificateIssuer Structure

C/C++

```
struct I_KS_CertificateIssuer {
  I_T_CHAR * dist_name_p;
  I_T_CHAR * alt_name_p;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Certificate Issuer Structure.

5.4.30 I_KS_CertificateSubject Structure

C/C++

```
struct I_KS_CertificateSubject {
  I_T_CHAR * dist_name_p;
  I_T_CHAR * alt_name_p;
};
```

File

File: kmipkeymgmttypes.h

Description

Attribute Certificate Subject Structure.

5.4.31 I_KS_SupportedKMIPVersions Structure

C/C++

```
struct I_KS_SupportedKMIPVersions {
  I_T_CHAR ** supportedKmipVers_pp;
  I_T_UINT count;
};
```

File

File: kmipkeymgmttypes.h

Description

Structure containing array of Supported KMIP Versions.

5.5 KMIP Key Management Opaque Objects

This section describes the KMIP Key Management opaque objects available in CADP CAPI KMIP.

Types

Name	Description
I_KO_AttributeList	An opaque object representing an attribute list.
BatchHandle	Batch Object Structure.

5.5.1 I_KO_AttributeList Type

C/C++

```
typedef struct _KO_AttributeList * I_KO_AttributeList;
```

File

File: kmipkeymgmttypes.h

Description

An opaque object representing an attribute list.

5.5.2 BatchHandle Type

C/C++

```
typedef struct _BatchObjectStruct * BatchHandle;
```

File

File: kmipkeymgmttypes.h

Description

Batch Object Structure.

5.6 Key Management Macros

This section describes the KMIP Key Management macros available in CADP CAPI KMIP.

Macros

Name	Description
I_KT_RESET_ALL_OPERATIONS	For this value of flag, it deletes the entire list of operations including the operations list, input list, and output list etc. Therefore, the status of the batch is set to empty.
I_KT_RESET_OUTPUT_LIST	For this value of flag, it flushes the output, but keeps the input argument and operation list that can be reused for another execution in the batch.
_KMIPERR_H	This is macro _KMIPERR_H.
KEYFINDNAME	This is macro KEYFINDNAME.

5.6.1 I_KT_RESET_ALL_OPERATIONS Macro

C/C++

```
#define I_KT_RESET_ALL_OPERATIONS 0x00000002 // For this value of flag, it deletes
the entire list of operations including the operations list, input list, and output
list etc. Therefore, the status of the batch is set to empty.
```

File

File: kmipkeymgmttypes.h

Description

For this value of flag, it deletes the entire list of operations including the operations list, input list, and output list etc. Therefore, the status of the batch is set to empty.

5.6.2 I_KT_RESET_OUTPUT_LIST Macro

C/C++

```
#define I_KT_RESET_OUTPUT_LIST 0x00000001 // For this value of flag, it flushes the output, but keeps the input argument and operation list that can be reused for another execution in the batch.
```

File

File: kmipkeymgmttypes.h

Description

For this value of flag, it flushes the output, but keeps the input argument and operation list that can be reused for another execution in the batch.

5.6.3 _KMIPERR_H Macro

C/C++

```
#define _KMIPERR_H
```

File

File: kmiperr.h

Description

This is macro _KMIPERR_H.

5.6.4 KEYFINDNAME Macro

C/C++

```
#define KEYFINDNAME
```

File

File: cadp_capi_naekeymgmt.h

Description

This is macro KEYFINDNAME.

6 Cryptographic API

This chapter describes the Cryptographic API.

When Symmetric Cache is enabled, the Cryptographic functions perform local encryption using the keys that are fetched from CipherTrust Manager by using either the NAE XML protocol or KMIP protocol.

When Symmetric Cache is disabled, the Cryptographic functions perform remote cryptographic operations by using the NAE XML protocol.

The I_C_CryptBulk function work only in the remote cryptographic mode using the NAE XML protocol.

Note: CADP CAPI client cannot offload cryptographic operations to CipherTrust Manager.

6.1 Cryptographic Functions

This section describes the Cryptographic functions available in CADP CAPI KMIP.

Functions

	Name	Description
	I_C_CalculateEncipheredSize	
	I_C_CalculateEncipheredSizeForKey	
	I_C_CalculateOutputSize	This function calculates the size of the output text given the cipher and the input size. **Note:** When using the RSA keys of size greater than 2048 bits, I_C_CalculateOutputSize returns the incorrect required output size.
	I_C_CalculateOutputSizeForKey	This function calculates the size of output text (plus any key metadata) given the cipher and the input size. This function supports Versioned keys. **Note:** When using the RSA keys of size greater than 2048 bits, I_C_CalculateOutputSizeForKey returns the incorrect required output size.
	I_C_CryptFinal	This function receives the last chunk of data and is used with I_C_CryptInit/I_C_CryptUpdate. Use the I_C_CryptInit/I_C_CryptUpdate/I_C_CryptFinal interface (multiple updates are OK) when you want results back from part of your cryptographic operation before you have all the data ready, or if your data is larger than I_C_Crypt will allow. I_C_CryptUpdate and I_C_CryptFinal block while waiting for the results.
	I_C_DeleteCipherSpec	This function deletes an I_O_CipherSpec object. You must call this function after you are done using the I_O_CipherSpec object.

	I_C_DeleteEVP	This function deletes EVP Context saved. Use I_C_DeleteEVP to delete EVP context saved. This function should be called only when I_C_Crypt_Fast is used by the user. Since EVP Context is saved only when I_C_Crypt_Fast is called.
	I_C_DeleteUserSpec	This function deletes an I_O_UserSpec object. You must call this function after you are done using the I_O_UserSpec object.
	I_C_GetCipherBlockSize	This function returns the block size of a cipher in bytes. Ciphers that use RC4, RSA, or HMAC algorithms will always return a block size of zero, as they are not block ciphers. **Note:** This function assesses the block size of the algorithm associated with the cipher object, it does *not* verify that the cipher object's algorithm and key name are correct.
	I_C_StateClear	This function free the memory allocated to I_O_CipherState structure. This API will be called only after calling I_C_Crypt_Fast to free the state structure.
	I_C_StateInit	This function initializes I_O_CipherState structure. This structure holds connection object which holds EVP context. This API will be called only before calling I_C_Crypt_Fast.

6.1.1 I_C_CalculateEncipheredSize Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CalculateEncipheredSize(I_O_CipherSpec cipher, I_T_Operation
operation, I_T_UINT plaintextSize, I_T_UINT * ciphertextSize);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_CipherSpec cipher	The cipher that will be applied to the plaintext.
I_T_Operation operation	The cryptographic operation that will be performed on the plaintext.
I_T_UINT plaintextSize	The unsigned integer that holds the plaintext size.
I_T_UINT * ciphertextSize	The unsigned integer that will hold the ciphertext size.

Notes

This function is deprecated. Use I_C_CalculateOutputSize(). The I_C_CalculateEncipheredSize function calculates the size of a ciphertext given the cipher and the plaintext size.

6.1.2 I_C_CalculateEncipheredSizeForKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CalculateEncipheredSizeForKey(I_O_Session session,
I_O_CipherSpec cipher, I_T_Operation operation, I_T_UINT plaintextSize, I_T_UINT *
ciphertextSize);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be applied to the plaintext.
I_T_Operation operation	The cryptographic operation that will be performed on the plaintext.
I_T_UINT plaintextSize	The unsigned integer that holds the plaintext size.
I_T_UINT * ciphertextSize	The unsigned integer that will hold the ciphertext size.

Notes

This function is deprecated. Use I_C_CalculateOutputSizeForKey(). The I_C_CalculateEncipheredSizeForKey function calculates the size of a ciphertext (plus any key metadata) given the cipher and the plaintext size.

6.1.3 I_C_CalculateOutputSize Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CalculateOutputSize(I_O_CipherSpec cipher, I_T_Operation
operation, I_T_UINT inputSize, I_T_UINT * outputSize);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_CipherSpec cipher	The cipher to be applied to the input text.
I_T_Operation operation	The cryptographic operation to be performed on the input text.
I_T_UINT inputSize	The unsigned integer that holds the input size.
I_T_UINT * outputSize	The unsigned integer that holds the output size.

Description

This function calculates the size of the output text given the cipher and the input size.

Note: When using the RSA keys of size greater than 2048 bits, I_C_CalculateOutputSize returns the incorrect required output size.

6.1.4 I_C_CalculateOutputSizeForKey Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CalculateOutputSizeForKey(I_O_Session session, I_O_CipherSpec
```

```
cipher, I_T_Operation operation, I_T_UINT inputSize, I_T_UINT * outputSize);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherSpec cipher	The cipher that will be applied to the input text.
I_T_Operation operation	The cryptographic operation that will be performed on the input text.
I_T_UINT inputSize	The unsigned integer that holds the input size.
I_T_UINT * outputSize	The unsigned integer that will hold the output size.

Description

This function calculates the size of output text (plus any key metadata) given the cipher and the input size. This function supports Versioned keys.

Note: When using the RSA keys of size greater than 2048 bits, I_C_CalculateOutputSizeForKey returns the incorrect required output size.

6.1.5 I_C_CryptFinal Function

C/C++

```
I_T_RETURN FUNCEXP I_C_CryptFinal(I_O_Session session, I_O_CipherState state,
I_T_BYTE * outData, I_T_UINT * outDataLen);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_Session session	The current session.
I_O_CipherState state	An object containing information about the cryptographic operation, such as bytes sent.
I_T_BYTE * outData	The resulting data.
I_T_UINT * outDataLen	The length of the output data.

Description

This function receives the last chunk of data and is used with I_C_CryptInit/I_C_CryptUpdate.

Use the I_C_CryptInit/I_C_CryptUpdate/I_C_CryptFinal interface (multiple updates are OK) when you want results back from part of your cryptographic operation before you have all the data ready, or if your data is larger than I_C_Crypt will allow. I_C_CryptUpdate and I_C_CryptFinal block while waiting for the results.

6.1.6 I_C_DeleteCipherSpec Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DeleteCipherSpec(I_O_CipherSpec cipher);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_CipherSpec cipher	The cipher object that will be deleted.

Description

This function deletes an I_O_CipherSpec object. You must call this function after you are done using the I_O_CipherSpec object.

6.1.7 I_C_DeleteEVP Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DeleteEVP(I_O_CipherState * state);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_CipherState * state	The cipher state structure which holds connection object which contains information about the cryptographic operation such as bytes sent.

Description

This function deletes EVP Context saved.

Use I_C_DeleteEVP to delete EVP context saved. This function should be called only when I_C_Crypt_Fast is used by the user. Since EVP Context is saved only when I_C_Crypt_Fast is called.

6.1.8 I_C_DeleteUserSpec Function

C/C++

```
I_T_RETURN FUNCEXP I_C_DeleteUserSpec(I_O_UserSpec userSpec);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
userspec	The userspec object that will be deleted.

Description

This function deletes an I_O_UserSpec object. You must call this function after you are done using the I_O_UserSpec object.

6.1.9 I_C_GetCipherBlockSize Function

C/C++

```
I_T_RETURN FUNCEXP I_C_GetCipherBlockSize(I_O_CipherSpec cipher, I_T_UINT *
blockSize);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_CipherSpec cipher	The cipher that will be analyzed.
I_T_UINT * blockSize	The unsigned integer to hold the block size.

Description

This function returns the block size of a cipher in bytes. Ciphers that use RC4, RSA, or HMAC algorithms will always return a block size of zero, as they are not block ciphers.

Note: This function assesses the block size of the algorithm associated with the cipher object, it does *not* verify that the cipher object's algorithm and key name are correct.

6.1.10 I_C_StateClear Function

C/C++

```
I_T_RETURN FUNCEXP I_C_StateClear(I_O_CipherState * state);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_CipherState * state	The cipher state structure which holds connection object which contains information about the cryptographic operation such as bytes sent.

Description

This function free the memory allocated to I_O_CipherState structure. This API will be called only after calling I_C_Crypt_Fast to free the state structure.

6.1.11 I_C_StateInit Function

C/C++

```
I_T_RETURN FUNCEXP I_C_StateInit(I_O_CipherState * state);
```

File

File: cadp_capi_crypto.h

Parameters

Parameters	Description
I_O_CipherState * state	The cipher state structure which holds connection object which contains information about the cryptographic operation such as bytes sent.

Description

This function initializes I_O_CipherState structure. This structure holds connection object which holds EVP context. This API will be called only before calling I_C_Crypt_Fast.

6.2 Cryptographic Enumerations

This section describes the Cryptographic enumerations available in CADP CAPI KMIP.

Enumerations

	Name	Description
	I_T_IVTypeEnum	The IV type describes how the IV will be applied for bulk operations. Use the I_T_IVType typedef in your code.
	I_T_OperationEnum	The operation type describes the cryptographic operation to be performed. Use the I_T_Operation typedef in your code.
	I_T_UserSpec_param	enum of UserSpec parameter , addtional user input in crpto operation
	I_T_UspecTypeEnum	The USPEC type describes how the User spec will be applied for bulk operations. Use the I_T_UspecType typedef in your code.
	I_T_FpeFormat_Type_param	enum of FPE Format type parameter , user input format type for FPE crypto operation

6.2.1 I_T_IVTypeEnum Enumeration

C/C++

```
enum I_T_IVTypeEnum {
  I_T_IV_PerElement = 0,
  I_T_IV_Single = 1,
  I_T_IV_None = 2
};
```

File

File: cadp_capi_crypto.h

Members

Members	Description
I_T_IV_PerElement = 0	There will be one IV for each element.
I_T_IV_Single = 1	A single IV will be applied to all of the elements.
I_T_IV_None = 2	No IV is applied.

Description

The IV type describes how the IV will be applied for bulk operations. Use the I_T_IVType typedef in your code.

6.2.2 I_T_OperationEnum Enumeration

C/C++

```
enum I_T_OperationEnum {
  I_T_Operation_Encrypt = 0,
  I_T_Operation_Decrypt = 1,
  I_T_Operation_PublicEncrypt = 2,
  I_T_Operation_PrivateDecrypt = 5,
  I_T_Operation_MAC = 7,
  I_T_Operation_MACV = 8,
  I_T_Operation_Sign = 9,
  I_T_Operation_SignV = 10
};
```

File

File: cadp_capi_crypto.h

Members

Members	Description
I_T_Operation_Encrypt = 0	Encrypt plaintext.
I_T_Operation_Decrypt = 1	Decrypt ciphertext.
I_T_Operation_PublicEncrypt = 2	Encrypt plaintext with the public portion on an RSA key.
I_T_Operation_PrivateDecrypt = 5	Decrypt ciphertext with the private portion of an RSA key.
I_T_Operation_MAC = 7	Create a message authentication code (MAC).

I_T_Operation_MACV = 8	Verify a message authentication code (MAC).
I_T_Operation_Sign = 9	Create a digital signature.
I_T_Operation_SignV = 10	Verify a digital signature.

Description

The operation type describes the cryptographic operation to be performed. Use the I_T_Operation typedef in your code.

6.2.3 I_T_UserSpec_param Enumeration

C/C++

```
enum I_T_UserSpec_param {
  I_T_USPEC_TWEAKALGO = 0,
  I_T_USPEC_TWEAKDATA = 1,
  I_T_USPEC_AUTHTAG = 2,
  I_T_USPEC_AADDATA = 3,
  I_T_USPEC_AUTHTAGLEN = 4,
  I_T_USPEC_AUTHTAG_VERIFY = 5,
  I_T_USPEC_FORMAT = 6,
  I_T_USPEC_CERTLIST = 7,
  I_T_USPEC_EC_EPHEMERAL_KEY = 8,
  I_T_USPEC_EC_MAC_DATA = 9,
  I_T_USPEC_EC_ENCRYPTION = 10,
  I_T_USPEC_RADIX = 11,
  I_T_USPEC_CHARSET = 12,
  I_T_USPEC_UTFMODE = 13,
  I_T_USPEC_VERSION_HEADER_MODE = 14,
  I_T_USPEC_VERSION_HEADER = 15,
  I_T_USPEC_CHARSET_RANGE = 16,
  I_T_USPEC_CARD62_CHARSET_ORDER = 17,
  I_T_USEPC_NONE
};
```

File

File: cadp_capi_crypto.h

Members

Members	Description
I_T_USPEC_TWEAKALGO = 0	Tweakalgo in userpec
I_T_USPEC_TWEAKDATA = 1	TweakData in userpec
I_T_USPEC_AUTHTAG = 2	Auth Tag in userspec
I_T_USPEC_AADDATA = 3	AAD Data in userspec
I_T_USPEC_AUTHTAGLEN = 4	Auth Tag Len in userspec
I_T_USPEC_AUTHTAG_VERIFY = 5	Auth Tag verification in userspec
I_T_USPEC_FORMAT = 6	CMS format used for sign verify
I_T_USPEC_CERTLIST = 7	List of CA used in sign verify CMS format
I_T_USPEC_EC_EPHEMERAL_KEY = 8	to set and get Ephemeral key in userspec
I_T_USPEC_EC_MAC_DATA = 9	to set and get MAC Data in userspec
I_T_USPEC_EC_ENCRYPTION = 10	Userspec creation for EC

I_T_USPEC_RADIX = 11	Radix in userpec for FPE/FF3 and FF1
I_T_USPEC_CHARSET = 12	charset in userpec for FPE/FF3 and FF1
I_T_USPEC_UTFMODE = 13	Utf mode in userpec for FPE/FF3 and FF1
I_T_USPEC_VERSION_HEADER_MODE = 14	Type of version header(NONE/Internal/External) in userpec for FPE
I_T_USPEC_VERSION_HEADER = 15	Key version header in userpec for FPE/FF3 and FF1
I_T_USPEC_CHARSET_RANGE = 16	charset range in userpec for FPE/FF3 and FF1, FF1v2
I_T_USPEC_CARD62_CHARSET_ORDER = 17	charset order sequence of CARD62 in userpec for FPE/FF3 and FF1, FF1v2
I_T_USEPC_NONE	No param selected

Description

enum of UserSpec parameter , addtional user input in crpto operation

6.2.4 I_T_UspecTypeEnum Enumeration

C/C++

```
enum I_T_UspecTypeEnum {
  I_T_Uspec_PerElement = 0,
  I_T_Uspec_Single = 1,
  I_T_Uspec_None = 2
};
```

File

File: cadp_capi_crypto.h

Members

Members	Description
I_T_Uspec_PerElement = 0	There will be one Uspec for each element.
I_T_Uspec_Single = 1	A single Uspec will be applied to all of the elements.
I_T_Uspec_None = 2	No Uspec is applied.

Description

The USPEC type describes how the User spec will be applied for bulk operations. Use the I_T_UspecType typedef in your code.

6.2.5 I_T_FpeFormat_Type_param Enumeration

C/C++

```
enum I_T_FpeFormat_Type_param {
  I_T_NONE = 0,
  I_T_LAST_FOUR = 1,
  I_T_FIRST_SIX_LAST_FOUR = 2,
  I_T_FIRST_SIX = 3,
  I_T_FIRST_TWO_LAST_FOUR = 4
```

```
};
```

File

File: cadp_capi_crypto.h

Members

Members	Description
I_T_NONE = 0	NO FORMAT
I_T_LAST_FOUR = 1	last 4 token
I_T_FIRST_SIX_LAST_FOUR = 2	first 6 and last 4 tokens
I_T_FIRST_SIX = 3	first 6 token
I_T_FIRST_TWO_LAST_FOUR = 4	first 2 and last 4 token

Description

enum of FPE Format type parameter , user input format type for FPE crypto operation

6.3 Cryptographic Opaque Objects

This section describes the Cryptographic opaque objects available in CADP CAPI KMIP.

Types

Name	Description
I_O_CipherSpec	An opaque object representing a reusable algorithm specification
I_O_CipherState	An opaque object representing a given encryption operation
I_O_StringList	An opaque object representing a string list
I_O_UserSpec	An opaque object representing a given user input parameters

6.3.1 I_O_CipherSpec Type

C/C++

```
typedef struct _O_CipherSpec * I_O_CipherSpec;
```

File

File: cadp_capi_crypto.h

Description

An opaque object representing a reusable algorithm specification

6.3.2 I_O_CipherState Type

C/C++

```
typedef struct _O_CipherState * I_O_CipherState;
```

File

File: cadp_capi_crypto.h

Description

An opaque object representing a given encryption operation

6.3.3 I_O_StringList Type

C/C++

```
typedef struct _O_StringList * I_O_StringList;
```

File

File: cadp_capi_naekeymgmt.h

Description

An opaque object representing a string list

6.3.4 I_O_UserSpec Type

C/C++

```
typedef struct _O_UserSpec * I_O_UserSpec;
```

File

File: cadp_capi_crypto.h

Description

An opaque object representing a given user input parameters

6.4 Cryptographic Macros

This section describes the cryptographic macros available in CADP CAPI KMIP.

Macros

Name	Description
I_T_LNG_ALG_AES_CBC_NOPADDING	AES CBC mode with NoPadding

I_T_LNG_ALG_AES_CBC_PKCS5PADDING	AES CBC mode with PKCS5Padding
I_T_LNG_ALG_AES_ECB_NOPADDING	AES ECB mode with NoPadding
I_T_LNG_ALG_AES_ECB_PKCS5PADDING	AES ECB mode with PKCS5Padding
I_T_LNG_ALG_DES_CBC_NOPADDING	DES CBC mode with NoPadding
I_T_LNG_ALG_DES_CBC_PKCS5PADDING	DES CBC mode with PKCS5Padding
I_T_LNG_ALG_DES_ECB_NOPADDING	DES ECB mode with NoPadding
I_T_LNG_ALG_DES_ECB_PKCS5PADDING	DES ECB mode with PKCS5Padding
I_T_LNG_ALG_DES_EDE_CBC_NOPADDING	DESede CBC mode with NoPadding
I_T_LNG_ALG_DES_EDE_CBC_PKCS5PADDING	DESede CBC mode with PKCS5Padding
I_T_LNG_ALG_DES_EDE_ECB_NOPADDING	DESede ECB mode with NoPadding
I_T_LNG_ALG_DES_EDE_ECB_PKCS5PADDING	DESede ECB mode with PKCS5Padding
I_T_LNG_ALG_HMACSHA1	HMAC SHA1
I_T_LNG_ALG_HMACSHA256	HMAC SHA256
I_T_LNG_ALG_HMACSHA384	HMAC SHA384
I_T_LNG_ALG_HMACSHA512	HMAC SHA512
I_T_LNG_ALG_RC4	RC4
I_T_LNG_ALG_RSA	RSA
I_T_LNG_ALG_SEED	SEED
I_T_LNG_ALG_SHA1WITHRSA	SHA1 RSA
I_T_LNG_ALG_SHA256WITHRSA	SHA256 RSA
I_T_LNG_ALG_SHA384WITHRSA	SHA384 RSA
I_T_LNG_ALG_SHA512WITHRSA	SHA512 RSA
I_T_MAX_BULK_DATA_SIZE	This is macro I_T_MAX_BULK_DATA_SIZE.

6.4.1 I_T_LNG_ALG_AES_CBC_NOPADDING Macro

C/C++

```
#define I_T_LNG_ALG_AES_CBC_NOPADDING "AES/CBC/NoPadding"
```

File

File: cadp_capi_crypto.h

Description

AES CBC mode with NoPadding

6.4.2 I_T_LNG_ALG_AES_CBC_PKCS5PADDING Macro

C/C++

```
#define I_T_LNG_ALG_AES_CBC_PKCS5PADDING "AES/CBC/PKCS5Padding"
```

File

File: cadp_capi_crypto.h

Description

AES CBC mode with PKCS5Padding

6.4.3 I_T_LNG_ALG_AES_ECB_NOPADDING Macro

C/C++

```
#define I_T_LNG_ALG_AES_ECB_NOPADDING "AES/ECB/NoPadding"
```

File

File: cadp_capi_crypto.h

Description

AES ECB mode with NoPadding

6.4.4 I_T_LNG_ALG_AES_ECB_PKCS5PADDING Macro

C/C++

```
#define I_T_LNG_ALG_AES_ECB_PKCS5PADDING "AES/ECB/PKCS5Padding"
```

File

File: cadp_capi_crypto.h

Description

AES ECB mode with PKCS5Padding

6.4.5 I_T_LNG_ALG_DES_CBC_NOPADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_CBC_NOPADDING "DES/CBC/NoPadding"
```

File

File: cadp_capi_crypto.h

Description

DES CBC mode with NoPadding

6.4.6 I_T_LNG_ALG_DES_CBC_PKCS5PADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_CBC_PKCS5PADDING "DES/CBC/PKCS5Padding"
```

File

File: cadp_capi_crypto.h

Description

DES CBC mode with PKCS5Padding

6.4.7 I_T_LNG_ALG_DES_ECB_NOPADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_ECB_NOPADDING "DES/ECB/NoPadding"
```

File

File: cadp_capi_crypto.h

Description

DES ECB mode with NoPadding

6.4.8 I_T_LNG_ALG_DES_ECB_PKCS5PADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_ECB_PKCS5PADDING "DES/ECB/PKCS5Padding"
```

File

File: cadp_capi_crypto.h

Description

DES ECB mode with PKCS5Padding

6.4.9 I_T_LNG_ALG_DES_EDE_CBC_NOPADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_EDE_CBC_NOPADDING "DESede/CBC/NoPadding"
```

File

File: cadp_capi_crypto.h

Description

DESede CBC mode with NoPadding

6.4.10 I_T_LNG_ALG_DES_EDE_CBC_PKCS5PADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_EDE_CBC_PKCS5PADDING "DESede/CBC/PKCS5Padding"
```

File

File: cadp_capi_crypto.h

Description

DESede CBC mode with PKCS5Padding

6.4.11 I_T_LNG_ALG_DES_EDE_ECB_NOPADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_EDE_ECB_NOPADDING "DESede/ECB/NoPadding"
```

File

File: cadp_capi_crypto.h

Description

DESede ECB mode with NoPadding

6.4.12 I_T_LNG_ALG_DES_EDE_ECB_PKCS5PADDING Macro

C/C++

```
#define I_T_LNG_ALG_DES_EDE_ECB_PKCS5PADDING "DESede/ECB/PKCS5Padding"
```

File

File: cadp_capi_crypto.h

Description

DESede ECB mode with PKCS5Padding

6.4.13 I_T_LNG_ALG_HMACSHA1 Macro

C/C++

```
#define I_T_LNG_ALG_HMACSHA1 "HmacSHA1"
```

File

File: cadp_capi_crypto.h

Description

HMAC SHA1

6.4.14 I_T_LNG_ALG_HMACSHA256 Macro

C/C++

```
#define I_T_LNG_ALG_HMACSHA256 "HmacSHA256"
```

File

File: cadp_capi_crypto.h

Description

HMAC SHA256

6.4.15 I_T_LNG_ALG_HMACSHA384 Macro

C/C++

```
#define I_T_LNG_ALG_HMACSHA384 "HmacSHA384"
```

File

File: cadp_capi_crypto.h

Description

HMAC SHA384

6.4.16 I_T_LNG_ALG_HMACSHA512 Macro

C/C++

```
#define I_T_LNG_ALG_HMACSHA512 "HmacSHA512"
```

File

File: cadp_capi_crypto.h

Description

HMAC SHA512

6.4.17 I_T_LNG_ALG_RC4 Macro

C/C++

```
#define I_T_LNG_ALG_RC4 "RC4"
```

File

File: cadp_capi_crypto.h

Description

RC4

6.4.18 I_T_LNG_ALG_RSA Macro

C/C++

```
#define I_T_LNG_ALG_RSA "RSA"
```

File

File: cadp_capi_crypto.h

Description

RSA

6.4.19 I_T_LNG_ALG_SEED Macro

C/C++

```
#define I_T_LNG_ALG_SEED "SEED"
```

File

File: cadp_capi_crypto.h

Description

SEED

6.4.20 I_T_LNG_ALG_SHA1WITHRSA Macro

C/C++

```
#define I_T_LNG_ALG_SHA1WITHRSA "SHA1withRSA"
```

File

File: cadp_capi_crypto.h

Description

SHA1 RSA

6.4.21 I_T_LNG_ALG_SHA256WITHRSA Macro

C/C++

```
#define I_T_LNG_ALG_SHA256WITHRSA "SHA256withRSA"
```

File

File: cadp_capi_crypto.h

Description

SHA256 RSA

6.4.22 I_T_LNG_ALG_SHA384WITHRSA Macro

C/C++

```
#define I_T_LNG_ALG_SHA384WITHRSA "SHA384withRSA"
```

File

File: cadp_capi_crypto.h

Description

SHA384 RSA

6.4.23 I_T_LNG_ALG_SHA512WITHRSA Macro

C/C++

```
#define I_T_LNG_ALG_SHA512WITHRSA "SHA512withRSA"
```

File

File: cadp_capi_crypto.h

Description

SHA512 RSA

6.4.24 I_T_MAX_BULK_DATA_SIZE Macro

C/C++

```
#define I_T_MAX_BULK_DATA_SIZE 100;
```

File

File: cadp_capi_crypto.h

Description

This is macro I_T_MAX_BULK_DATA_SIZE.