# KMIP – Key Management Interoperability Protocol

Draft Version 0.98

Last revision February 10th, 2009

Editor: Robert Haas, IBM Zurich Research Laboratory

# Table of Contents



 

 

 

 

 

# 1 Introduction

This document is intended as a specification of the protocol used for the communication between clients and servers to perform certain management operations on objects stored and maintained by a key management system. These objects will be referred to as *Managed Objects* in this specification. They include symmetric and asymmetric cryptographic keys, digital certificates, and templates used to simplify the creation of objects and control their use. Managed Objects are managed with *operations* that include the ability to generate cryptographic keys, register objects with the key management system, obtain objects from the system, destroy objects from the system, and search for objects maintained by the system. Managed Objects also have associated *attributes,* which are named values stored by the key management system and which can be obtained from the system via operations. Certain attributes may be changed, added or deleted, again by operations.

The protocol specified in this document includes several certificate-related functions for which there are a number of existing protocols – namely Validate (e.g. SVP or XKMS), Certify (e.g. CMP, CMC, SCEP) and Re-certify (e.g. CMP, CMC, SCEP). The protocol does not attempt to define a comprehensive certificate management protocol such as would be required for a certification authority. However, it does include functions that are needed in proxying certificate management functions through a key server.

In addition to the normative definitions for managed objects, operations and attributes, this specification also includes normative definitions for the following aspects of the protocol:

- Message contents and formats
- Authentication profiles for clients and servers
- Message encoding, including enumerations
- Error handling

This specification is complemented by two other documents. The Usage Guide provides illustrative information on using the protocol. The Test Specification provides samples of protocol messages corresponding to a set of defined test cases.

# 2 Objects

The following subsections describe the objects that are passed between the clients and servers of the key management system. Some of these object types, called *Base Objects*, are used only in the protocol itself, and are not considered Managed Objects. Key management systems may choose to support a subset of the Managed Objects. The object descriptions refer to the primitive data types they are composed of. These primitive data types are

- Integer
- Long Integer
- Big Integer
- Enumeration – choices from a predefined list of values
- Boolean
- Text String – string of characters representing human-readable text
- Octet String – sequence of unencoded byte values
- Date-Time – date and time, with a granularity of one second
- Interval – time interval expressed in seconds

Structures are composed of ordered lists of primitive data types or structures.

 

# 1 Base Objects

These objects are used within the messages of the protocol, but are not objects managed by the key management system. They may be components of Managed Objects.

## 1 Attribute

An object, used for sending and receiving Managed Object attributes. The *Attribute Name* is a text-string which is used to identify the attribute. The *Attribute Index* is an index number assigned by the key management server when a specified named attribute is allowed to have multiple instances. The index number is used to identify the particular instance. Index numbers start with 0. The index number of an attribute is never changed when other instances are added or deleted. For example, if a particular attribute has 4 instances with index numbers 0, 1, 2 and 3, and the instance with index 2 is deleted, the index number of instance 3 is not changed. Attributes which have a single instance have an Attribute Index of 0, which is assumed if the index is not specified. The *Attribute Value* is either a primitive data type, or structured object, depending on the attribute.

| Object | Encoding | Required |
|---|---|---|
| Attribute | Structure | Yes |
| Attribute Name | Text String | Yes |
| Attribute Index | Integer | No |
| Attribute Value | Varies, depending on attribute. See Section 3 | Yes |

## 2 Credential

A credential is a protocol-only object, used for client identification purposes and not managed by the key management system, e.g., user id/password pairs, Kerberos tokens, etc. See Section 8 .

| Object | Encoding | Required |
|---|---|---|
| Credential | Structure | Yes |
| Credential Type | Enumeration | Yes |
| Credential Value | Octet String | Yes |

## 3 Key Block

A *Key Block* object encapsulates all of the information that is closely associated with a cryptographic key. A Key Block object may contain different information depending on who it is sent to and when it is sent. It contains a Key Value of one of the following *Key Value Types*:

- *Raw* – This is a key which consists of "pure" cryptographic key material, encoded as a string of bytes.
- *Opaque* – This is an encoded key for which the encoding is unknown to the key management system. It is encoded as a string of bytes.
- *PKCS1* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#1 object.
- *PKCS8* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#8 object, supporting both RSAPrivateKey syntax and EncryptedPrivateKey.
- Several *Transparent Key* types – These are algorithm-specific structures containing defined values for the various key types, as defined in Section 2.1.6 .

 

- *Extensions* – These are vendor-specific extensions to allow for proprietary or legacy key formats.

It contains also the Cryptogtaphic Algorithm and the Cryptographic Length. Some example values are:

- RSA keys are typically 1024, 2048 or 3072 bits in length
- 3DES keys are typically 168 bits in length
- AES keys are typically 128 or 256 bits in length

The Key Block may optionally contain a Key Wrapping Data structure, which indicates that the key is wrapped, or MAC'ed/signed, or both.

| Object | Encoding | Required Field |
|---|---|---|
| Key Block | Structure | Yes |
| Key Value Type | Enumeration | Yes |
| Key Value | Octet String: for wrapped Key Value; Structure: for plaintext Key Value | Yes |
| Cryptographic Algorithm | Enumeration | Yes, may be omitted only if this information is encapsulated in the Key Value. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Length below must also be present. |
| Cryptographic Length | Integer | Yes, may be omitted only if this information is encapsulated in the Key Value. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Algorithm above must also be present. |
| Key Wrapping Data | Structure | No |

## 4 Key Value

The *Key Value* is used only inside a Key Block and is either an Octet String or a structure:

- The Key Value structure contains the key material, either as an octet string or as a Transparent Key structure (see Section 2.1.7 ), and optional attribute information that is associated with and encapsulated with the key material. This attribute information differs from the attributes associated with Managed Objects, and which is obtained via the Get Attributes operation, only by the fact that it is encapsulated with, and may be wrapped, signed or MAC'ed along with the key material itself.
- The Key Value Octet String is the wrapped TTLV-encoded (see Section 9.1 ) Key Value structure.

| Object | Encoding | Required Field |
|---|---|---|
| Key Value | Structure | Yes |
| Key Material | Octet String: for Raw, Opaque, PKCS1, PKCS8, or Vendor Extension Key Value types; Structure: for Transparent, or Vendor Extension Key Value | Yes |

 

| | | |
|---|---|---|
| | Types | |
| Attribute | Attribute Object, see Section 2.1.1 | No. May be repeated |

## 5 Key Wrapping Data

The Key Block may also supply optional information about a cryptographic key wrapping mechanism used to wrap the Key Value. This consists of a *Key Wrapping Data* structure.

This structure contains:

- A *Wrapping Method* that indicates the method used to wrap the Key Value.
- An *Encryption Key Information* with the Unique Identifier value for the encryption key.
- A *MAC/Signature Key Information* with the Unique Identifier value for the MAC'ing or signing key.
- A *MAC/Signature* field with the MAC or signature of the Key Value.
- An *IV/Counter/Nonce* if required by the wrapping method.

If wrapping is used, the whole Key Value structure is wrapped with the wrapping key material unless otherwise specified by the Wrapping Method. The algorithm is determined by the Cryptographic Algorithm attribute set for the key. Similarly, the Cryptographic Parameters attribute of the key will identify the mode of operation or hashing algorithm to be used.

The following wrapping methods are currently defined:

- *Encrypt* only (possibly includes authenticated encryption algorithms that use a single key)
- *MAC/sign* only
- *Encrypt then MAC/sign*
- *MAC/sign then encrypt*
- *TR-31*
- *Extensions*

| Object | Encoding | Required Field |
|---|---|---|
| Key Wrapping Data | Structure | Yes |
| Wrapping Method | Enumeration | Yes |
| Encryption Key Information | Structure | No |
| MAC/Signature Key Information | Structure | No. Corresponds to the symmetric key used to MAC the Key Value or the private key used to sign the Key Value |
| MAC/Signature | Octet String | No |
| IV/Counter/Nonce | Octet String | No |

The structures of the Encryption Key Information and the MAC/Signature Key Information are as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Encryption Key Information | Structure | Yes |
| Unique Identifier | Text string | Yes |

 

| | | |
|---|---|---|
| Cryptographic Parameters | Structure | No |

| Object | Encoding | Required Field |
|---|---|---|
| MAC/Signature Key Information | Structure | Yes |
| Unique Identifier | Text string | Yes. It can be the Unique Identifier of the Private or of the Public Key |
| Cryptographic Parameters | Structure | No |

## 6 Key Wrapping Specification

This is a separate structure defined for operations that provide the option to return wrapped keys. The *Key Wrapping Specification* must be specified inside the operation request, if clients wish the server to return a wrapped key. If Cryptographic Parameters are specified in the Encryption Key Information and the MAC/Signature Key Information, then the server can verify that they match one of the instances of the Cryptographic Parameters attribute of the corresponding key. If Cryptographic Parameters are omitted, the server can choose to use the Cryptographic Parameters attribute with the lowest index of the corresponding key. If the corresponding key does not have any Cryptographic Parameters attribute, or if no match is found, an error is returned.

This structure contains :

- A Wrapping Method that indicates the method used to wrap the Key Value.
- An Encryption Key Information with the Unique Identifier value of the encryption key and associated cryptographic parameters.
- A MAC/Signature Key Information with the Unique Identifier value of the MAC'ing or signing key and associated cryptographic parameters.
- Zero or more Attribute Names to indicate the attributes to be wrapped with the key material.

| Object | Encoding | Required Field |
|---|---|---|
| Key Wrapping Specification | Structure | Yes |
| Wrapping Method | Enumeration | Yes |
| Encryption Key Information | Structure | No |
| MAC/Signature Key Information | Structure | No |
| Attribute Name | Text String | No, May be repeated |

The structures of the Encryption Key Information and the MAC/Signature Key Information are defined in Section 2.1.5 .

## 7 Transparent Key Structures

*Transparent Key* structures describe key material in a form that can easily be interpreted by all participants in the protocol. They are used in the Key Value structure.

 

### *1 Transparent Symmetric Key*

If the Key Value Type in the Key Block is *Transparent Symmetric Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| Key | Octet String | Yes |

### *2 Transparent DSA Private Key*

If the Key Value Type in the Key Block is *Transparent DSA Private Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| P | Big Integer | Yes |
| Q | Big Integer | Yes |
| G | Big Integer | Yes |
| X | Big Integer | Yes |

### *3 Transparent DSA Public Key*

If the Key Value Type in the Key Block is *Transparent DSA Public Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| P | Big Integer | Yes |
| Q | Big Integer | Yes |
| G | Big Integer | Yes |
| Y | Big Integer | Yes |

### *4 Transparent RSA Private Key*

If the Key Value Type in the Key Block is *Transparent RSA Private Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| Modulus | Big Integer | Yes |
| Private Exponent | Big Integer | No |
| Public Exponent | Big Integer | No |
| P | Big Integer | No |
| Q | Big Integer | No |
| Prime Exponent P | Big Integer | No |
| Prime Exponent Q | Big Integer | No |

 

| | | |
|---|---|---|
| CRT Coefficient | Big Integer | No |

Note: One of the following must be present:

- Private Exponent
- P and Q
- Prime Exponent P and Prime Exponent Q.

### *5 Transparent RSA Public Key*

If the Key Value Type in the Key Block is *Transparent Rsa Public Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| Modulus | Big Integer | Yes |
| Public Exponent | Big Integer | Yes |

### *6 Transparent DH Private Key*

If the Key Value Type in the Key Block is *Transparent DH Private Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| P | Big Integer | Yes |
| G | Big Integer | Yes |
| Q | Big Integer | No |
| J | Big Integer | No |
| X | Big Integer | Yes |

Note: Q=P-1, J where P=JQ+1

### *7 Transparent DH Public Key*

If the Key Value Type in the Key Block is *Transparent DH Public Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| P | Big Integer | Yes |
| G | Big Integer | Yes |
| Q | Big Integer | No |
| J | Big Integer | No |
| Y | Big Integer | Yes |

$Y=G^X \bmod P$, Q=P-1, J where P=JQ+1

### *8 Transparent ECDSA Private Key*

If the Key Value Type in the Key Block is *Transparent ECDSA Private Key*, then Key Material is a

 

structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| Recommended Curve | Enumeration | Yes |
| D | Big Integer | Yes |

### *9 Transparent ECDSA Public Key*

If the Key Value Type in the Key Block is *Transparent ECDSA Public Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| Recommended Curve | Enumeration | Yes |
| Q String | Octet String | Yes |

### *10 Transparent ECDH Private Key*

If the Key Value Type in the Key Block is *Transparent ECDH Private Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| Recommended Curve | Enumeration | Yes |
| D | Big Integer | Yes |

### *11 Transparent ECDH Public Key*

If the Key Value Type in the Key Block is *Transparent ECDH Public Key*, then Key Material is a structure as follows:

| Object | Encoding | Required Field |
|---|---|---|
| Key Material | Structure | Yes |
| Recommended Curve | Enumeration | Yes |
| Q String | Octet String | Yes |

## 8 Template-Attribute Structures

These structures are used in various operations to provide the desired attributes values and/or template names in the request and to return the actual attributes values in the response.

The *Template-Attribute*, *Common Template-Attribute*, *Private Key Template-Attribute,* and *Public Key Template-Attribute* structures are defined identically as follows:

| Object | Encoding | Required Field |
|---|---|---|

 

| | | |
|---|---|---|
| Template-Attribute, Common Template-Attribute, Private Key Template-Attribute, Public Key Template-Attribute | Structure | Yes |
| Template Name | Text String | No, May be repeated. |
| Attribute | Attribute Object, see Section 2.1.1 | No, May be repeated |

The Template Name is the Name of a Template object or a Policy Template object, as defined in Sections 2.2.6 and 2.2.7 .

# 2 Managed Objects

Managed Objects are objects that are the subjects of key management operations, which are described in Section 4 . Managed Objects include all objects that may be registered with the system. *Managed Cryptographic Objects* are the subset of Managed Objects that contain cryptographic material, e.g. certificates, keys, and secret data. Managed Cryptographic Objects may have operations performed on them, and may have attributes that do not apply to all Managed Objects.

## 1 Certificate

A Managed Cryptographic Object, which is a digital certificate, such as an encoded X.509 certificate.

| Object | Encoding | Required Field |
|---|---|---|
| Certificate | Structure | Yes |
| Certificate Type | Enumeration | Yes |
| Certificate Value | Octet String | Yes |

## 2 Symmetric Key

A Managed Cryptographic Object, which is a symmetric key.

| Object | Encoding | Required Field |
|---|---|---|
| Symmetric Key | Structure | Yes |
| Key Block | Structure | Yes |

## 3 Public Key

A Managed Cryptographic Object, which is the public portion of an asymmetric key pair. This is a “raw” public key, not a certificate.

| Object | Encoding | Required Field |
|---|---|---|
| Public Key | Structure | Yes |
| Key Block | Structure | Yes |

## 4 Private Key

A Managed Cryptographic Object, which is a the private portion of an asymmetric key pair.

 

| Object | Encoding | Required Field |
|---|---|---|
| Private Key | Structure | Yes |
| Key Block | Structure | Yes |

## 5 Split Key

A Managed Cryptographic Object, which is a split key. A split key is a secret, usually a symmetric key or a private key that has been split into a number of parts, each of which can then be distributed to several key holders, for additional security. The *Split Key Parts* field contains the total number of parts, and the *Split Key Threshold* field contains the minimum number of parts needed to reconstruct the entire key. The *Key Part Identifier* indicates which key part is contained in the cryptographic object, and must be at least 1 and less than or equal to Split Key Parts.

| Object | Encoding | Required Field |
|---|---|---|
| Split Key | Structure | Yes |
| Split Key Parts | Integer | Yes |
| Key Part Identifier | Integer | Yes |
| Split Key Threshold | Integer | Yes |
| Split Key Method | Enumeration | Yes |
| Prime Field Size | Big Integer | No, required only if Split Key Method is Polynomial Sharing Prime Field. |
| Key Block | Structure | Yes |

There are three *Split Key Methods* for secret sharing: the first one is based on XOR and the other two are based on polynomial secret sharing, according to [Adi Shamir, "How to share a secret", Communications of the ACM, vol. 22, no. 11, pp. 612-613], as explained further in the Usage Guide. Let $L$ be the minimum number of bits needed to represent all values of the secret.

- When the Split Key Method is XOR, the Key Material in the Key Value of the Key Block is of length $L$ bits. The number of split keys is Split Key Parts (identical to Split Key Threshold), and the secret is reconstructed by XOR'ing all of them
- When the Split Key Method is Polynomial Sharing Prime Field, secret sharing is performed in the field GF(*Prime Field Size*), represented as integers, where Prime Field Size is a prime bigger than $2^L$.
- When the Split Key Method is Polynomial Sharing GF($2^{16}$), secret sharing is performed in the field GF($2^{16}$). The Key Material in the Key Value of the Key Block is a bit string of length $L$, and when $L$ is bigger than $2^{16}$ , then secret sharing is applied piecewise in pieces of 16 bits each. The Key Material in the Key Value of the Key Block is the concatenation of the corresponding shares of all pieces of the secret.

  Secret sharing is performed in the field GF($2^{16}$), which is represented as an algebraic extension of GF($2^8$):

  $$GF(2^{16}) \approx GF(2^8)\,[y]/(y^2+y+m), \quad \text{where } m \text{ is defined later.}$$

  An element of this field then consists of a linear combination $uy + v$, where $u$ and $v$ are elements of the smaller field GF($2^8$).

  The representation of field elements and the notation in this section rely on FIPS PUB 197, Sections 3 and 4. The field GF($2^8$) is as described in FIPS PUB 197,

  $$GF(2^8) \approx GF(2)\,[x]/(x^8+x^4+x^3+x+1).$$

 

An element of GF($2^8$) is represented as an octet. Addition and subtraction in GF($2^8$) can be performed as a bitwise XOR of the octets. Multiplication and inversion are more complex: see FIPS PUB 197 Section 4.1 and 4.2 for details.

An element of GF($2^{16}$) is represented as a pair of octets ($u$, $v$). The element $m$ is given by

$$m = x^5+x^4+x^3+x,$$

which is represented by the octet 0x3A (or {3A} in notation according to FIPS PUB 197).

Addition and subtraction in GF($2^{16}$) both correspond to simply XORing the octets. The product of two elements $ry + s$ and $uy + v$ is given by

$$(ry + s)\,(uy + v) = ((r + s)(u + v) + sv)y + (ru + svm).$$

The inverse of an element $uy + v$ is given by

$$(uy + v)^{-1} = ud^{-1}y + (u + v)d^{-1}, \text{ where } d = (u + v)v + mu^2.$$

## 6 Template

A Template is a named Managed Object containing the client-settable attributes of a Managed Cryptographic Object. It is essentially a stored, named list of attributes. A Template is used to specify the attributes of a new Managed Cryptographic Object in various operations. It is intended to be used to specify the cryptographic attributes of new objects in a standardized or convenient way. None of the attributes specified in a Template except the Name attribute apply to the template object itself, but instead apply to any object created or registered using the Template.

The Template may be the subject of the Register, Locate, Get, Get Attributes, Get Attribute List, Add Attribute, Modify Attribute, Delete Attribute, and Destroy operations.

The attributes that may be contained in a Template are:

- Name (This is the name of the Template, not the name of its target object.)
- Cryptographic Algorithm
- Cryptographic Length
- Object Group
- Application Specific Identification
- Contact Information
- Custom Attribute

| Object | Encoding | Required Field |
|---|---|---|
| Template | Structure | Yes |
| Attribute | Attribute Object, see Section 2.1.1 | Yes. May be repeated. |

## 7 Policy Template

A *Policy Template* is a named Managed Object containing attributes. The purpose of a Policy Template is to encapsulate all of the policy-related attributes into a Managed Object which may be independent of any single Managed Cryptographic Object, and may be managed and transmitted independently. Only policy-related attributes may be stored in a Policy Template. The Policy Template may be the subject of the Register, Locate, Get, get Attributes, Get Attribute List, Add Attribute, Modify Attribute, Delete Attribute, and Destroy operations. The attributes which may be contained in a Policy Template are:

 

- Name (This is the name of the Policy Template, not the name of any object to which it is applied.)
- Cryptographic Algorithm
- Cryptographic Parameters
- Operation Policy Name
- Cryptographic Usage Mask
- Usage Limits
- Activation Date
- Process Start Date
- Protect Stop Date
- Deactivation Date
- Custom Attribute

| Object | Encoding | Required Field |
|---|---|---|
| Policy Template | Structure | Yes |
| Attribute | Attribute Object, see Section 2.1.1 | Yes. May be repeated |

### 8 Secret Data

A Managed Cryptographic Object containing a shared secret that is not a key or certificate, e.g., a password. The Key Block used to contain *Secret Data* should contain a (possibly wrapped) Key Value of the Opaque type.

| Object | Encoding | Required Field |
|---|---|---|
| Secret Data | Structure | Yes |
| Secret Data Type | Enumeration | Yes |
| Key Block | Structure | Yes |

### 9 Opaque Object

A Managed Object that the key management server may not be able to interpret, but will store. The context information for this object can be stored and retrieved using Custom Attributes.

| Object | Encoding | Required Field |
|---|---|---|
| Opaque Object | Structure | Yes |
| Opaque Data Type | Enumeration | Yes |
| Opaque Data Value | Octet String | Yes |

# 3 Attributes

The following subsections describe the attributes that are associated with Managed Objects. These attributes may be obtained by a client from the server using the Get Attribute operation. Some attributes may be set by the Add Attribute operation or updated by the

 

Modify Attribute operation, and some may be deleted by the Delete Attribute operation if they no longer apply to the Managed Object.

When attributes are returned by the server, e.g. via a Get Attributes operation, the returned attribute value may differ depending on the client. For example, the Cryptographic Usage Mask value may be different for different clients, depending on the policy of the server. Similarly, when a client modifies an attribute, this is merely a mechanism for sending information to the server. The server may store the attribute as received, or modify the attribute before saving it, or combine it with information from other sources, or merely use it as advice on how to modify its internal knowledge of the cryptographic object. The choice depends on server functionality, policy, and the kind of attribute being modified.

The attribute name contained in the first row of the Object column of the first table in each subsection is the canonical name used when managing attributes using the Get Attributes, Get Attribute List, Add Attribute, Modify Attribute, and Delete Attribute operations.

The second table in each subsection lists certain attribute characteristics, such as "Must always have a value". The "When implicitly set" characteristic indicates which operations (other than operations that manage attributes) can implicitly result in adding or modifying the attribute of the object. They can be object(s) on which the operation is performed or object(s) created as a result of the operation. Implicit attribute changes occur even if the attribute is not specified in the operation request itself.

## 1 Unique Identifier

The *Unique Identifier* is generated by the key management system to uniquely identify a Managed Object. It is only required to be unique within the identifier space managed by a single key management system, however it is recommended that this identifier be globally unique, to allow for key management domain export of such objects. This attribute is assigned by the key management system at creation or registration time, and may never be changed or deleted by any entity at any time.

| Object | Encoding | Required Field |
|---|---|---|
| Unique Identifier | Text String | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

## 2 Name

The *Name* attribute is used to identify and locate the object, assigned by the client. The key management system may specify rules for valid names which may be created by the client. Clients will be informed of such rules by a mechanism which is not specified here. Names must be unique within a given key management domain, but are not required to be globally unique.

| Object | Encoding | Required Field |
|---|---|---|
| Name | Structure | Yes |
| Name Value | Text String | Yes |
| Name Type | Enumeration | Yes |

 

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Client |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Re-key, Re-certify |
| Applies to Object Types | All Objects |

## 3 Object Type

The type of a Managed Object, e.g. public key, private key, symmetric key, etc. This attribute is set by the server when the object is created or registered and is never changed.

| Object | Encoding | Required Field |
|---|---|---|
| Object Type | Enumeration | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

## 4 Cryptographic Algorithm

The cryptographic algorithm used by the object, e.g. RSA, DSA, DES, 3DES, AES, etc. This attribute is set by the server when the object is created or registered and is never changed.

| Object | Encoding | Required Field |
|---|---|---|
| Cryptographic Algorithm | Enumeration | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |

 

| | |
|---|---|
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Re-key |
| Applies to Object Types | All Cryptographic Objects |

# 5 Cryptographic Length

*Cryptographic Length* is the length in bits of the cryptographic key material of the Managed Cryptographic Object. This attribute is set by the server when the object is created or registered, and is never changed.

| Object | Encoding | Required Field |
|---|---|---|
| Cryptographic Length | Integer | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Re-key |
| Applies to Object Types | All Cryptographic Objects |

# 6 Cryptographic Parameters

The *Cryptographic Parameters* attribute is a structure that contains a set of optional fields that describe certain cryptographic parameters to be used when performing cryptographic operations using the object. Specific fields may only pertain to certain types of Managed Cryptographic Objects.

| Object | Encoding | Required Field |
|---|---|---|
| Cryptographic Parameters | Structure | Yes |
| Block Cipher Mode | Enumeration | No |
| Padding Method | Enumeration | No |
| Hashing Algorithm | Enumeration | No |
| Role Type | Enumeration | No |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |

 

| | |
|---|---|
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Re-key, Re-certify |
| Applies to Object Types | All Cryptographic Objects |

Role Types are defined as follows:

| |
|---|
| ZMK – Shared key to allow transfer of subordinate keys between two entities |
| ZPK – Shared key to allow transfer of PINs between two entities |
| MAC – MAC key, specifically X9.9/19 retail MAC |
| CVK – Key for generating/verifying 3-digit VISA/Mastercard signature strip codes (CVV/CVC) |
| CSC – Key for generating/verifying 4-digit American Express Card Security Codes |
| PVKIBM – Derivation key for derived PINs checked with the IBM offset method |
| PVKPVV – Verification key for random PINs checked with the PVV method |
| MKCVC – Master key for dynamic CVC calculations |
| MKSMI – Master key for smart card secure messaging integrity |
| MKSMC – Master key for smart card secure messaging confidentiality |
| MKIDN – Master key for Card Dynamic Number |
| MKAC – Master key for Chip card cryptogram |
| MKCAP – Master key for Cardholder Authentication Programme |
| BDK – Base derivation key for DUKPT |

# 7 Certificate Type

The type of a certificate, e.g. X.509, PGP, etc. This value is set by the server when the certificate is created or registered and is never changed.

| Object | Encoding | Required Field |
|---|---|---|
| Certificate Type | Enumeration | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |

 

| | |
|---|---|
| When implicitly set | Create, Register, Certify, Re-certify |
| Applies to Object Types | Certificates |

# 8 Certificate Issuer

An identification of a certificate, containing the Issuer Distinguished Name (from the Issuer field of the certificate) and the Certificate Serial Number (from the Serial Number field of the certificate). This value is set by the server when the certificate is created or registered and is never changed.

| Object | Encoding | Required Field |
|---|---|---|
| Certificate Issuer | Structure | Yes |
| Issuer | Text String | Yes |
| Serial Number | Text String | Yes (for X.509 certificates) / No (for PGP certificates since they don't contain a serial number) |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Certify, Re-certify |
| Applies to Object Types | Certificates |

# 9 Certificate Subject

Identifies the subject of a certificate, containing the Subject Distinguished Name (from the Subject field of the certificate). It may optionally include one or more alternative names (e.g. email address, IP address, DNS name) for the subject of the certificate (from the Subject Alternative Name extension within the certificate). These values are set by the server when the certificate is created or registered and are not changed until the certificate is renewed.

It is possible to issue an X.509 certificate where the subject field is left blank as long as the Subject Alternative Name extension is included in the certificate and is marked *CRITICAL*. Therefore an empty string is an acceptable value for Certificate Subject.

| Object | Encoding | Required Field |
|---|---|---|
| Certificate Subject | Structure | Yes |
| Certificate Subject Distinguished Name | Text String | Yes |
| Certificate Subject Alternative Name | Text String | No, May be repeated |

 

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Register, Certify, Re-certify |
| Applies to Object Types | Certificates |

## 10 Digest

A digest of the key (digest of the Key Material), certificate (digest of the Certificate Value), or opaque object (digest of the Opaque Data Value). Multiple digests may be calculated using different algorithms. The mandatory digest is computed with the SHA-256 hashing algorithm, the server can store additional optional digests. The digest(s) are static and generated by the server when the object is created or registered.

| Object | Encoding | Required Field |
|---|---|---|
| Digest | Structure | Yes |
| Hashing Algorithm | Enumeration | Yes |
| Digest Value | Octet String | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | Yes |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Cryptographic Objects |

## 11 Operation Policy Name

An indication of what entities may perform which key management operations on the object. The contents of the *Operation Policy Name* attribute is the name of a policy object known to the key management system and therefore server dependent. The named policy objects are created and managed using mechanisms outside the scope of the protocol. The policies determine who may perform specified operations on the object, and which of the objects' attributes may be modified, or deleted, and by whom. It is expected that the Operation Policy Name attribute will be set when operations such as Create or

 

Register are executed. It is set either explicitly or via some default set by the server, and will then apply to all subsequent operations on the object.

| Object | Encoding | Required Field |
|---|---|---|
| Operation Policy Name | Text String | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

## 1 Operations outside of operation policy control

Some of the operations should be allowed to any client at any time, without respect to operation policy. These operations are:

- Create
- Create Key Pair
- Register
- Certify
- Validate
- Query
- Cancel
- Poll

## 2 Default Operation Policy

A key management system implementation should implement at least one named operation policy, which is used for objects where the *Operation Policy* attribute is not specified by the Client in a *Create* or *Register* operation, or in a template specified in these operations. This policy is named *default*. It specifies the following rules for operations on objects created or registered with this policy, depending on the object type.

### 1 *Default Operation Policy for Secret Objects*

This policy applies to Symmetric Keys, Private Keys, Split Keys, Secret Data, and Opaque Objects.

| Default Operation Policy for Secret Objects | |
|---|---|
| **Operation** | **Policy** |
| Re-Key | Allowed to creator only |

 

| | |
|---|---|
| Derive Key | Allowed to creator only |
| Locate | Allowed to creator only |
| Check | Allowed to creator only |
| Get | Allowed to creator only |
| Get Attributes | Allowed to creator only |
| Get Attribute List | Allowed to creator only |
| Add Attribute | Allowed to creator only |
| Modify Attribute | Allowed to creator only |
| Delete Attribute | Allowed to creator only |
| Obtain Lease | Allowed to creator only |
| Get Usage Allocation | Allowed to creator only |
| Activate | Allowed to creator only |
| Revoke | Allowed to creator only |
| Destroy | Allowed to creator only |
| Archive | Allowed to creator only |
| Recover | Allowed to creator only |

For mandatory profiles, the creator must be the transport-layer identification (see Usage Guide) provided at the Create or Register operation time.

### 2 *Default Operation Policy for Certificates and Public Key Objects*

This policy applies to Certificates and Public Keys.

| **Default Operation Policy for Certificates and Public Key Objects** | |
|---|---|
| **Operation** | **Policy** |
| Certify | Allowed to creator only |
| Re-certify | Allowed to creator only |
| Locate | Allowed to all |
| Check | Allowed to all |
| Get | Allowed to all |
| Get Attributes | Allowed to all |
| Get Attribute List | Allowed to all |
| Add Attribute | Allowed to creator only |
| Modify Attribute | Allowed to creator only |
| Delete Attribute | Allowed to creator only |
| Obtain Lease | Allowed to all |
| Activate | Allowed to creator only |
| Revoke | Allowed to creator only |
| Destroy | Allowed to creator only |

 

| | |
|---|---|
| Archive | Allowed to creator only |
| Recover | Allowed to creator only |

### *3 Default Operation Policy for Template and Policy Template Objects*

The operation policy specified as an attribute in the *Create* operation for a template object is the operation policy used for objects that will be created using that template, and is not the policy used to control operations on the template itself. There is no mechanism provided for specifying a policy used to control operations on template objects, so the default policy for template objects themselves is always used for templates created by clients using the *Register* operation to create template objects.

| **Default Operation Policy for Private Template Objects** | |
|---|---|
| **Operation** | **Policy** |
| Locate | Allowed to creator only |
| Get | Allowed to creator only |
| Get Attributes | Allowed to creator only |
| Get Attribute List | Allowed to creator only |
| Add Attribute | Allowed to creator only |
| Modify Attribute | Allowed to creator only |
| Delete Attribute | Allowed to creator only |
| Destroy | Allowed to creator only |

In addition to private template objects, which are controlled by the above policy which can be created by clients or the server, publicly known and usable templates may be created and managed by the server, with a different default policy for these template objects.

| **Default Operation Policy for Public Template Objects** | |
|---|---|
| **Operation** | **Policy** |
| Locate | Allowed to all |
| Get | Allowed to all |
| Get Attributes | Allowed to all |
| Get Attribute List | Allowed to all |
| Add Attribute | Disallowed to all |
| Modify Attribute | Disallowed to all |
| Delete Attribute | Disallowed to all |
| Destroy | Disallowed to all |

# 12 Cryptographic Usage Mask

The *Cryptographic Usage Mask* defines the cryptographic usage of a key. This is a bit mask which indicates to the client which cryptographic functions may be performed using the key.

- Sign

 

- Verify
- Encrypt
- Decrypt
- Wrap
- Unwrap
- Export
- MAC
- MAC Verify
- Derive Key
- Content Commitment
- Key Agreement
- Certificate Sign
- CRL Sign

This list takes into consideration values which may appear in the Key Usage extension in an X.509 certificate. However, the list does not consider the more fined grained usages which may appear in the Extended Key Usage extension.

X.509 Key Usage values shall be mapped to Cryptographic Usage Mask values in the following manner:

| **X.509 Key Usage to Cryptographic Usage Mask Mapping** | |
|---|---|
| **X.509 Key Usage Value** | **Cryptographic Usage Mask Value** |
| digitalSignature | Sign and Verify |
| contentCommitment | Content Commitment<br>(Non Repudiation) |
| keyEncipherment | Wrap and Unwrap |
| dataEncipherment | Encrypt and Decrypt |
| keyAgreement | Key Agreement |
| keyCertSign | Certificate Sign |
| cRLSign | CRL Sign |
| encipherOnly | Encrypt |
| decipherOnly | Decrypt |

The Content Commitment (Non-Repudiation) Cryptographic Usage Mask value shall be set for public keys used to verify digital signatures for non-repudiation purposes (to protect against a signing entity denying an action). Public keys used to verify digital signatures for other purposes such as authentication and integrity shall be set with the Sign, Verify or both Cryptographic Usage Mask values.

| **Object** | **Encoding** | **Required Field** |
|---|---|---|
| Cryptographic Usage Mask | Integer | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server or Client |
| Modifiable by server | Yes |

 

| | |
|---|---|
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Cryptographic Objects |

# 13 Lease Time

The *Lease Time* attribute defines a time interval for a Managed Object that indicates how long a client should use the object. This attribute always holds the initial value of a lease, and not the actual remaining time. Note that once the lease expires, the client must renew the lease by calling Obtain Lease. A server should store in this attribute the maximum Lease Time it is willing to serve and a client must request lease times (with Obtain Lease) which are less than, or equal. This attribute is read-only for clients. It can be modified by the server only.

| Object | Encoding | Required Field |
|---|---|---|
| Lease Time | Interval | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Keys |

# 14 Usage Limits

This is a mechanism for limiting the usage of a Managed Cryptographic Object. It only applies to Managed Cryptographic Objects that can be used for protection purposes (symmetric keys, private keys, public keys, etc.) and it must only reflect their usage for protection (encryption, signing, etc.). This attribute may not exist for all Managed Cryptographic Objects, since some objects may be used without limit, depending on client/server policies. Usage for process purposes (decryption, verification, etc.) is not limited. The attribute has four fields for two different types of limits. Exactly one of these two types (either bytes or objects) must be present. These limits are:

- *Usage Limits Total Bytes* – the total number of bytes allowed to be protected. This is the total value for the entire life of the object, and is never changed once the object begins to be used for protection purposes.
- *Usage Limits Total Objects* – the total number of objects allowed to be protected. This is the total value for the entire life of the object, and is never changed once the object begins to be used for protection purposes.
- *Usage Limits Byte Count* – the currently remaining number of bytes allowed to be protected.

 

- *Usage Limits Object Count* – the currently remaining number of objects allowed to be protected.

When the attribute is initially set, usually during object creation or registration, the values set are the Total values allowed for the useful life of the object. The count values must be ignored by the server if the attribute is specified in a operation that creates a new object. Changes made via the Modify Attribute operation reflect corrections to these Total values, but they cannot be changed once the count values have changed by a Get Usage Allocation operation. The count values cannot be set or modified by the client via the Add Attribute or Modify Attribute operations.

| Object | Encoding | Required Field |
|---|---|---|
| Usage Limits | Structure | Yes |
| Usage Limits Total Bytes | Big Integer | No. Must be present if Usage Limits Byte Count is present |
| Usage Limits Total Objects | Big Integer | No. Must be present if Usage Limits Object Count is present |
| Usage Limits Byte Count | Big Integer | No. May only be present if Usage Limits Object Count is not present |
| Usage Limits Object Count | Big Integer | No. May only be present if Usage Limits Byte Count is not present |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Re-key, Get Usage Allocation |
| Applies to Object Types | Symmetric Keys, Private Keys, Split Keys, Public Keys |

# 15 State

This attribute is an indication of the state of an object as known to the key management server. The state may not be changed by using the Modify Attribute operation on this attribute. The state may only be changed by the server as a side effect of other operations or other server processes. An object may be in one of the following states at any given time. (Note: These states correspond to those described in NIST Special Publication 800-57).

- *Pre-Active*: The object exists but is not yet usable for any cryptographic purpose.
- *Active*: The object may be used for all cryptographic purposes which are allowed by its Cryptographic Usage Mask attribute.
- *Deactivated*: The object may not be used for protection purpose, e.g. encryption or signing, but, if permitted by the Cryptographic Usage Mask attribute, may be used for process purposes, e.g. decryption or verification, but only under extraordinary circumstances and when special permission is granted.

 

- *Compromised*: The object may have been compromised, and may only be used for process purposes in a client that is trusted to handle compromised cryptographic objects.
- *Destroyed*: The object is no longer usable for any purpose.
- *Destroyed Compromised*: The object is no longer usable for any purpose, however its compromised status may be retained for audit or security purposes.

State transitions occur as follows:

1. The transition from a non-existent key to Pre-Active is determined by the creation of the object. When an object is created or registered, it automatically goes from non-existent to Pre-Active. If, however, the operation that creates or registers the object contains an Activation Date that has already occurred, the state immediately transitions to Active. In this case, the server may set the Activation Date attribute to the time when the operation is received, depending on server policy. If the operation contains an Activation Date attribute in the future, or contains no Activation Date, it becomes initialized in the key management system in the Pre-Active state.
2. The transition from Pre-Active to Destroyed cannot be directly made by request of a client. The client may issue a Destroy operation. This will allow the server to destroy the object at the server's discretion.
3. The transition from Pre-Active to Compromised is performed by a client issuing a Revoke operation with a Revocation Reason of Compromised.
4. The transition from Pre-Active to Active can occur in one of two ways:
    - The object has an Activation Date in the future. At the time the Activation Date is reached, the server may change the state to Active.
    - A client issues a Modify Attribute operation, modifying the Activation Date to a date in the past, or the current date. In this case, the server may set the Activation Date attribute to the time when the operation that created or registered the object was received, depending on server policy.
    - A client issues an Activate operation on the object. The server will set the Activation Date to the time the Activate operation is received.
5. The transition from Active to Compromised is performed by a client issuing a Revoke operation with a Revocation Reason of Compromised.
6. The transition from Active to Deactivated can occur in one of two ways:
    - The object's Deactivation Date is reached. The server may change the state to Deactivated.
    - A client issues a Revoke operation, with a Revocation Reason other than Compromised.
    - The client issues a Modify Attribute operation, modifying the Deactivation Date to a date in the past, or the current date. In this case, the server may set the Deactivation Date attribute to the date in the past or the current date, depending on server policy.
7. The transition from Deactivated to Destroyed is requested by a client issuing a Destroy operation. The server will destroy the object when and if server policy dictates.
8. The transition from Deactivated to Compromised is performed by a client issuing a Revoke operation with a Revocation Reason of Compromised.
9. The transition from Compromised to Destroyed Compromised is requested by a client issuing a Destroy operation. The server will destroy the object when and if server policy dictates.
10. The transition from Destroyed to Destroyed Compromised is performed by a client issuing a *R*evoke operation with a Revocation Reason of Compromised.

Only the transitions described above are permitted.

| Object | Encoding | Required Field |
|---|---|---|

 

| | | |
|---|---|---|
| State | Enumeration | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Activate, Revoke, Destroy, Certify, Re-certify, Re-key |
| Applies to Object Types | All Cryptographic Objects |

## 16 Initial Date

The date and time when the Managed Cryptographic Object was first created or registered at the server. This time corresponds to state transition 1 (see Section 3.15 ). This attribute is set by the server when the object is created or registered, and is never changed. This attribute is also set for non-cryptographic objects (e.g. templates) when then are first registered with the server.

| Object | Encoding | Required Field |
|---|---|---|
| Initial Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | Yes |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

## 17 Activation Date

The date and time when the Managed Cryptographic Object may begin to be used. This time corresponds to state transition 4 (see Section 3.15 ). The object may not be used for any cryptographic purpose before the *Activation Date* has been reached. Once the state transition has occurred, this attribute may no longer be modified by the server or client. If a client attempts to set this value to a time in the past, the server may set it to the current time instead, depending on server policy.

| Object | Encoding | Required Field |
|---|---|---|
| Activation Date | Date-Time | Yes |

 

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Activate Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

## 18 Process Start Date

The date and time when a Managed Symmetric Key Object may begin to be used for process purposes, e.g. decryption or unwrapping, depending on the value of its Cryptographic Usage Mask attribute. The object may not be used for these cryptographic purposes before the *Process Start Date* has been reached. This value may be equal to, but may not precede, Activation Date. Once the Process Start Date has occurred, this attribute may no longer be modified by the server or the client. If a client attempts to set this value to a time in the past, the server may set it to the current time instead, depending on server policy.

| Object | Encoding | Required Field |
|---|---|---|
| Process Start Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Re-key |
| Applies to Object Types | Symmetric Keys |

## 19 Protect Stop Date

The date and time when a Managed Symmetric Key Object may no longer be used for protect purposes, e.g. encryption or wrapping, depending on the value of its Cryptographic Usage Mask attribute. This value may be equal to, but may not be later than Deactivation Date. Once the *Protect Stop Date* has occurred, this attribute may no longer be modified by the server or the client. If a client attempts to set this value to a time in the past, the server may set it to the current time instead, depending on server policy.

| Object | Encoding | Required Field |
|---|---|---|

 

| | | |
|---|---|---|
| Protect Stop Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Re-key |
| Applies to Object Types | Symmetric Keys |

## 20 Deactivation Date

The date and time when the Managed Cryptographic Object may no longer be used for any purpose, except for decryption, signature verification, or unwrapping, but only under extraordinary circumstances and when special permission is granted. This time corresponds to state transition 6 (see Section 3.15 ). Once this transition has occurred, this attribute may no longer be modified by the server or client. If a client attempts to set this value to a time in the past, the server may set it to the current time instead, depending on server policy.

| Object | Encoding | Required Field |
|---|---|---|
| Deactivation Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server or Client |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Revoke Certify, Re-certify, Re-key |
| Applies to Object Types | All Cryptographic Objects |

## 21 Destroy Date

The date and time when the Managed Cryptographic Object was destroyed. This time corresponds to state transitions 2, 7, or 9 (see Section 3.15 ). This value is set by the server when the object is destroyed due to reception of a Destroy operation, or due to server policy or out-of-band administrative action.

| Object | Encoding | Required Field |
|---|---|---|
| Destroy Date | Date-Time | Yes |

 

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Destroy |
| Applies to Object Types | All Objects |

## 22 Compromise Occurrence Date

The date and time when the Managed Cryptographic Object was first believed to be compromised. If it is not possible to estimate when the compromise occurred, this value should be set to the Initial Date for the object.

| Object | Encoding | Required Field |
|---|---|---|
| Compromise Occurrence Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Revoke |
| Applies to Object Types | Symmetric Keys, Private Keys, Split Keys, Secret Data, Opaque Object |

## 23 Compromise Date

The date and time when the Managed Cryptographic Object is entered into the compromised state. This time corresponds to state transitions 3, 5, 8, or 10 (see Section 3.15 ). This time represents when the key management system was made aware of the compromise, not necessarily when the compromise occurred. This attribute is set by the server when it receives a Revoke operation with a Revocation Reason of Compromised.

| Object | Encoding | Required Field |
|---|---|---|
| Compromise Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | No |

 

| | |
|---|---|
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Revoke |
| Applies to Object Types | Symmetric Keys, Private Keys, Split Keys, Secret Data, Opaque Object |

## 24 Revocation Reason

An indication of why the Managed Cryptographic Object was revoked, e.g. “compromised”, “expired”, “no longer used”, etc. This attribute is only changed by the server as a side effect of the Revoke Operation.

The *Revocation Message* is an optional field which is used exclusively for audit trail/logging purposes and may contain additional information about why the object was revoked, for example “Laptop stolen”, or “Machine decommissioned”.

| Object | Encoding | Required Field |
|---|---|---|
| Revocation Reason | Structure | Yes |
| Revocation Reason Code | Enumeration | Yes |
| Revocation Message | Text String | No |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Revoke |
| Applies to Object Types | All Cryptographic Objects |

## 25 Archive Date

The date and time when the Managed Object was placed in archival storage. This value is set by the server as a side effect of the Archive operation. This attribute is deleted whenever a Recover operation is performed.

| Object | Encoding | Required Field |
|---|---|---|
| Archive Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |

 

| | |
|---|---|
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Archive |
| Applies to Object Types | All Objects |

# 26 Object Group

An object may be part of a group of objects. An object may belong to more than one group. To assign an object to a group, the group name should be set into this attribute. The key management system may specify rules for the valid group names which may be created by the client. Clients will be informed of such rules by a mechanism which is not specified by this standard. In the protocol, the group names themselves are character strings of no specified format. Specific key management system implementations may choose to support hierarchical naming schemes or other syntax restrictions on the names. Groups may be used to associate objects for a variety of purposes. A set of keys used for a common purpose, but for different time intervals, may be linked by a common Object Group. Servers may create predefined groups and add objects to them independently of client requests.

| Object | Encoding | Required Field |
|---|---|---|
| Object Group | Text String | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

# 27 Link

A link from a Managed Cryptographic Object to another, closely related target Managed Cryptographic Object. The link has a type and the allowed types differ depending on the Object Type of the Managed Cryptographic Object. The *Linked Object Identifier* identifies the target Managed Cryptographic Object by its Unique Identifier. The link can contain such information as the private key corresponding to a public key, the parent certificate for a certificate in a chain, or for a derived symmetric key, the base key from which it was derived.

Possible values of *Link Type* in accordance with the Object Type of the Managed Cryptographic Object are:

- *Private Key Link.* For a Public Key object: the private key corresponding to the public key
- *Public Key Link.* For a Private Key object: the public key corresponding to the private key. For a Certificate object: the public key certified by the certificate
- *Certificate Link*. For Certificate objects: the parent certificate for a certificate in a certificate chain. For Public Key objects: the corresponding certificate(s), containing the same public key

 

- *Derivation Base Object Link* for a derived Symmetric Key object: the object(s) from which the current symmetric key was derived
- *Derived Key Link*: the symmetric key(s) that were derived from the current object.
- *Replacement Object Link.* For a Symmetric Key, Private Key, or Public Key object: the key that resulted from the re-key of the current key. For a Certificate object: the certificate that resulted from the re-certify. Note there can only be one such replacement object.
- *Replaced Object Link.* For a Symmetric Key, Private Key, or Public Key object: the key that was re-keyed to obtain the current key. For a Certificate object: the certificate that was re-certified to obtain the current certificate

The Link attribute should be present for private keys and public keys for which a certificate chain is stored by the server, and for certificates in a certificate chain.

Note that a Managed Object may have a Link attribute which has multiple values. For example, a Private Key may have links to the associated certificate as well as the associated public key. As another example, a Certificate object may have a Link attribute value to both the public key and to the certificate of the certification authority which signed the certificate.

It is also possible that a Managed Object does not have Link attribute values for associated cryptographic objects. This can occur in cases where the associated key material is not available to the server or client (consider the registration of a CA Signer certificate with a server but the corresponding private key is held in a different manner).

| Object | Encoding | Required Field |
|---|---|---|
| Link | Structure | Yes |
| Link Type | Enumeration | Yes |
| Linked Object Identifier | Text String | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Create Key Pair, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

# 28 Application Specific Identification

The *Application Specific Identification* is used to specify the intended use of a Managed Object. It consists two parts: the application name space that the object will be used with, and an identification specific to that application name space. The application name spaces are arbitrary text strings so that new types of application identifiers can be used without requiring the standard to be updated.

Some examples of application name space and identifier pairs:

- SMIME, 'someuser@company.com'
- SSL, 'some.domain.name'
- Volume Identification, '123343434'

 

- File Name, 'secret.doc'

The following application names spaces are recommended:

- SMIME
- SSL
- IPSEC
- HTTPS
- PGP
- Volume Identification
- File Name

Other values may be used according to server policy. No extension mechanism is defined or needed as any text string is allowable.

| Object | Encoding | Required Field |
|---|---|---|
| Application Specific Identification | Structure | Yes |
| Application Name Space | Text String | Yes |
| Application Identifier | Text String | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Client |
| Modifiable by server | No |
| Modifiable by client | Yes |
| Deletable by client | Yes |
| Multiple instances permitted | Yes |
| When implicitly set | Re-key, Re-certify |
| Applies to Object Types | All Cryptographic Objects |

# 29 Contact Information

The *Contact Information* attribute is optional and its content is used for contact purposes only. It is not used for policy enforcement. The attribute is set by the client or the server.

| Object | Encoding | Required Field |
|---|---|---|
| Contact Information | Text String | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes |
| Modifiable by client | Yes |
| Deletable by client | Yes |

 

| | |
|---|---|
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

## 30 Last Changed Date

A meta attribute that contains the date and time of the last change to the contents or attributes of the specified object.

| Object | Encoding | Required Field |
|---|---|---|
| Last Changed Date | Date-Time | Yes |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Server |
| Modifiable by server | Yes |
| Modifiable by client | No |
| Deletable by client | No |
| Multiple instances permitted | No |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Activate, Revoke, Destroy, Archive, Recover, Certify, Re-certify, Re-key, Get Usage Allocation |
| Applies to Object Types | All Objects |

## 31 Custom Attribute

A *Custom Attribute* is user-defined attribute and intended for vendor-specific purposes. It is created by the client and not interpreted by the server, or created by the server and either understood or not understood by the client. All custom attributes created by the client must adhere to a naming scheme where the name of the attribute must have a prefix of 'x-', meaning extended. The key management server may create and manage custom attributes which have a prefix of 'y-'. The tag type Custom Attribute cannot identify the particular attribute, hence such an attribute can only appear in an Attribute Structure with its name as defined in Section 2.1.1 .

| Object | Encoding | Required Field |
|---|---|---|
| Custom Attribute | Any data type or structure | Yes. The name of the attribute must start with 'x-' or 'y-'. |

| | |
|---|---|
| Must always have a value | No |
| Initially set by | Client or Server |
| Modifiable by server | Yes, for server-created attributes |

 

| | |
|---|---|
| Modifiable by client | Yes, for client-created attributes |
| Deletable by client | Yes, for client-created attributes |
| Multiple instances permitted | Yes |
| When implicitly set | Create, Create Key Pair, Register, Derive Key, Activate, Revoke, Destroy, Certify, Re-certify, Re-key |
| Applies to Object Types | All Objects |

# 4 Client-to-Server Operations

The following subsections describe the operations that may be requested by a key management client. Not all clients have to be capable of issuing all operation requests; however any client that issues a specific request must be capable of understanding the response to the request. All Object Management operations are sent in requests from clients to servers, and in responses from servers to clients. These operations may be combined into a batch, which allows multiple operations to be contained in a single request/response message pair.

A number of the operations whose descriptions follow are affected by a mechanism referred to as the *ID Placeholder*.

The key management server must implement a temporary variable called the ID Placeholder. This value consists of a single Unique Identifier. It is a variable stored inside the server that is only valid and preserved during the execution of a batch of operations. Once the batch of operations has been completed, the ID Placeholder value is discarded and/or invalidated by the server, so that subsequent requests will not find this previous ID Placeholder available.

The ID Placeholder is obtained from the Unique Identifier returned by the Create, Create Pair, Register, Derive Key, Re-Key, Certify, Re-Certify, Locate, and Recover operations. If any of these operations successfully completes and returns a Unique Identifier, then the server must copy this Unique Identifier into the ID Placeholder variable, where it is held until the completion of the operations remaining in the batched request. Subsequent operations in the batched request that need a Unique Identidier may make use of the ID Placeholder. This is indicated by omitting the Unique Identifier field from the request payloads for these operations. This mechanism is only valid if the Batch Error Continuation Option is set to Stop and the Batch Order Option is set to true.

Requests may contain attribute values to be assigned to the object. This information is specified with a Template-Attribute (see Section 2.1.8 ) that contains zero or more template names and zero or more individual attributes. If more than one template is specified, and there is a conflict between the single-value attributes in the templates, the value in the subsequent template takes precedence. If there is a conflict between the single-value attributes in the request and the single-value attributes in a specified template, the attribute values in the request take precedence. For multi-value attributes, the union of attribute values is used when the attributes are specified more than once.

Responses may contain attribute values that have been set differently than specified in the request. This information is specified with a Template-Attribute that contains one or more individual attributes.

## 1 Create

This operation requests the server to generate a new key as a Managed Cryptographic Object. This operation is not used to create Template or Policy Template objects (see Register operation, Section 4.3 ).

The request contains information about the type of object being created, and some of the attributes to be assigned to the object, e.g. Cryptographic Algorithm, Cryptographic Length, etc. This information may be specified by the names of Template objects which already exist. The response contains the Unique Identifier of the created object. The server must copy the Unique Identifier returned by this operation into the ID Placeholder variable.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required** | **Description** |

 

| | Field | |
|---|---|---|
| Object Type | Yes | Determines the type of object to be created |
| Template-Attribute | Yes | Specifies desired object attributes using templates and/or as individual attributes |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Object Type | Yes | Type of object created |
| Unique Identifier | Yes | The Unique Identifier of the newly created object |
| Template-Attribute | No | A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |

The following attributes must be included in the Create request, either explicitly, or via specification of a template that contains the attribute.

| Attribute | Required |
|---|---|
| Cryptographic Algorithm | Yes |
| Cryptographic Usage Mask | Yes |

## 2 Create Key Pair

This operation requests the server to generate a new public/private key pair and register the two corresponding new Managed Cryptographic Objects.

The request contains attributes to be assigned to the objects, e.g. Cryptographic Algorithm, Cryptographic Length, etc. Attributes and Template Names can be specified for both keys at the same time, by specifying a Common Template-Attribute object in the request. Attributes not common to both keys (e.g., Name, Cryptographic Usage Mask) may be specified using the Private Key Template-Attribute and Public Key Template-Attribute objects in the request which take precedence over the Common Template-Attribute object. A Link Attribute is automatically created by the server for each object, pointing to the corresponding object. The response contains the Unique Identifiers of both created objects. The ID Placeholder value will be set to the Unique Identifier of the Private Key.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Common Template-Attribute | No | Specifies desired attributes in templates and/or as individual attributes that apply to both the Private and Public Key Objects |
| Private Key Template-Attribute | No | Specifies templates and/or attributes that apply to the Private Key Object. Order of |

 

| | | |
|---|---|---|
| | | precedence applies |
| Public Key Template-Attribute | No | Specifies templates and/or attributes that apply to the Public Key Object. Order of precedence applies |

For multi-valued attributes, the union of the values found in the templates and attributes of the Common, Private , and Public Key Template-Attribute is used. For single-valued attributes, the order of precedence is as follows:

1. attributes specified explicitly in the Private and Public Key Template-Attribute, then
2. attributes specified via templates in the Private and Public Key Template-Attribute, then
3. attributes specified explicitly in the Common Template-Attribute, then
4. attributes specified via templates in the Common Template-Attribute

If there are multiple templates in the Common, Private, or Public Key Template-Attribute, then the subsequent value of the single-valued attribute takes precedence.

| **Response Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Private Key Unique Identifier | Yes | The Unique Identifier of the newly created Private Key object |
| Public Key Unique Identifier | Yes | The Unique Identifier of the newly created Public Key object |
| Private Key Template-Attribute | No | A list of attributes, for the Private Key Object, with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |
| Public Key Template-Attribute | No | A list of attributes, for the Public Key Object, with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |

The following attributes must be included and/or must have the same value in the *Create Key Pair* operation, either explicitly, or via specification of a template that contains the attribute.

| **Attribute** | **Required** | **Must contain the same value for both Private and Public Key** |
|---|---|---|
| Cryptographic Algorithm | Yes | Yes |
| Cryptographic Length | Yes | Yes |
| Cryptographic Usage Mask | Yes | No |
| Cryptographic Parameters | No | Yes |

 

| | | |
|---|---|---|
| Contact Information | No | Yes |

# 3 Register

This operation requests the server to register a Managed Object (created by the client or obtained by the client through some other means), allowing the server to manage the object. The arguments in the request are similar to those in the Create operation, but also may contain the object itself, for storage by the server. Optionally, objects which the client does not wish to be stored by the key management system may be omitted from the request, for example, private keys.

The request contains information about the type of object being registered, and some of the attributes to be assigned to the object, e.g. Cryptographic Algorithm, Cryptographic Length, etc. This information may be specified by the use of a Template-Attribute object. The response contains the Unique Identifier assigned by the server to the registered object. The server must copy the Unique Identifier returned by this operations into the ID Placeholder variable. The Initial Date attribute of the object is set to the current time.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Object Type | Yes | Determines the type of object being registered |
| Template-Attribute | Yes | Specifies desired object attributes using templates and/or as individual attributes |
| Certificate, Symmetric Key, Private Key, Public Key, Split Key, Secret Data or Opaque Object | No | The cryptographic object being registered. The object and attributes may be wrapped. Some objects, e.g. Private Keys, may be omitted from the request |

| **Response Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Object Type | Yes | Type of object registered |
| Unique Identifier | Yes | The Unique Identifier of the newly registered object |
| Template-Attribute | No | A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |

If the Register operation is being used to register a new Template, or Policy Template, then the request payload will contain a single Template Name field, containing the name of the new template, and the Cryptographic Object field must be omitted. The contents of the new Template or Policy Template will be the attributes contained in the Template-Attribute object in the request.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required** | **Description** |

 

| | Field | |
|---|---|---|
| Object Type | Yes | Template or Policy Template |
| Template Name | Yes | Specifies the name of the Template being registered |
| Template-Attribute | Yes, May be repeated | Specifies the attributes of the new Template using templates and/or as individual attributes |

When registering a new Template or a Policy Template, the attributes that may be included in the request are specified in Section 2.2.6 and 2.2.7 , respectively (note however that the Name attribute may not be specified). For all other object types that can be registered, the following attributes must be included in the Register request, either explicitly, or via specification of a template that contains the attribute.

| Attribute | Required |
|---|---|
| Cryptographic Algorithm | Yes, may be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Length below must also be present. |
| Cryptographic Length | Yes, may be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Algorithm above must also be present. |
| Cryptographic Usage Mask | Yes |

# 4 Re-key

This request is used to generate a replacement key for an existing symmetric key. It is analogous to the Create operation, except that many of the attributes of the new key are unchanged from the original key.

As the replacement key takes over the name attribute of the existing key, Re-key should only be performed once on a given key.

The server must copy the Unique Identifier of the replacement key returned by this operation into the ID Placeholder variable.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

As a result of Re-key, attributes of the existing key are changed similarly to performing a Revoke on that key with a Revocation Reason of Superseded, and the Link attribute is set to point to the replacement key.

If *Offset* is set, then the times of the new key will be set based on the times of the existing key (if such times exist) as follows:

| Attribute in Existing Key | Attribute in New Key |
|---|---|
| Initial Date ($IT_1$) | Initial Date ($IT_2$) > $IT_1$ |

 

| | |
|---|---|
| Activation Date ($AT_1$) | Activation Date ($AT_2$) = $IT_2$+ *Offset* |
| Process Start Date ($CT_1$) | Process Start Date ($CT_1$+($AT_2$- $AT_1$)) |
| Protect Stop Date ($TT_1$) | Protect Stop Date ($TT_1$+($AT_2$- $AT_1$)) |
| Deactivation Date ($DT_1$) | Deactivation Date ($DT_1$+($AT_2$- $AT_1$)) |

Attributes that are not copied from the existing key and are handled in a specific way are:

| Attribute | Action |
|---|---|
| Initial Date | Set to current time |
| Destroy Date | Not set |
| Compromise Occurrence Date | Not set |
| Compromise Date | Not set |
| Revocation Reason | Not set |
| Unique Identifier | New value generated |
| Usage Limits | The Total Bytes/Total Objects value is copied from the existing key, while the Byte Count/Object Count values are set to the Total Bytes/Total Objects. |
| Name | Set to the name(s) of the existing key; all name attributes of the existing key are removed. |
| State | Set based on attributes |
| Digest | Recomputed from the new key value |
| Link | Set to point to the existing key as the replaced key |
| Last Change Date | Set to current time |

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being re-keyed. If omitted, the ID Placeholder is substituted by the server |
| Offset | No | An Interval object indicating the difference between the Initialization Time of the new key and the Activation Date of the new key |
| Template-Attribute | No | Specifies desired object attributes using templates and/or as individual attributes |

 

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the new object |
| Template-Attribute | No | A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |

# 5 Derive Key

This request is used to derive a symmetric key using a key or secret that is already known to the key management system. It only applies to Managed Objects that can be used for key derivation (The Derive Key bit must be set in the Cryptographic Usage Mask attribute of the specified Managed Object). If the operation is issued for an object that does not have this bit set, the server must return a response with a Result Reason of Operation Not Supported. For all derivation methods, the client must specify the desired length of the derived key or secret using the Cryptographic Length attribute. If a key is created, the client must specify both the Cryptographic Length and Cryptographic Algorithm. If the specified length exceeds the output of the derivation method, the server must return an error. Clients have the option to derive multiple keys and IVs by creating a Secret Data object and specifying a Cryptographic Length that is the total length of the derived object. The length must not exceed the length of the output that is returned by the chosen derivation method.

The fields in the request specify the Unique Identifiers of the keys or secrets to be used for derivation (some derivation methods may require multiple keys or secrets to derive the result), the method to be used to perform the derivation, and any parameters needed by the specified method. The method is specified as an enumerated value. Currently defined derivation methods include:

- *PBKDF2* – This method is used to derive a symmetric key from a password or pass phrase. The PBKDF2 method is published in RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0, and also published as Internet Engineering Task Force's RFC 2898.
- *HASH* – This method derives a key by computing a hash over the derivation key or the derivation data.
- *HMAC* – This method derives a key by computing an HMAC over the derivation data.
- *ENCRYPT* – This method derives a key by encrypting the derivation data.
- *NIST800-108-C* – This method derives a key by computing the KDF in Counter Mode as specified in NIST SP 800-108.
- *NIST800-108-F* – This method derives a key by computing the KDF in Feedback Mode as specified in NIST SP 800-108.
- *NIST800-108-DPI* – This method derives a key by computing the KDF in Double-Pipeline Iteration Mode as specified in NIST SP 800-108.
- *Extensions*

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified. The server must perform the derivation function, and then register the derived object as a new Managed Object, returning the new Unique Identifier for the new object in the response. The server must copy the Unique Identifier returned by this operation into the ID Placeholder variable.

| Request Payload |
|---|

 

| Object | Required Field | Description |
|---|---|---|
| Object Type | Yes | Determines the type of object to be created |
| Unique Identifier | Yes. May be repeated | Determines the object or objects to be used to derive a new key from. At most two can be specified: one for the derivation key and another for the secret data. Note that the ID Placeholder cannot be used here. |
| Derivation Method | Yes | An Enumeration object specifying the method to be used to derive the new key |
| Derivation Parameters | Yes | A Structure object containing the parameters needed by the specified derivation method |
| Template-Attribute | Yes | Specifies desired object attributes using templates and/or as individual attributes; length must always be specified and algorithm is required for the creation of symmetric keys. |

| **Response Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Object Type | Yes | Type of object created |
| Unique Identifier | Yes | The Unique Identifier of the newly derived key |
| Template-Attribute | No, May be repeated | A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |

The *Derivation Parameters* for all derivation methods consist of the following parameters, except PBKDF2 that requires two additional parameters.

| Object | Encoding | Required Field |
|---|---|---|
| Derivation Parameters | Structure | Yes |
| Cryptographic Parameters | Structure | Yes, except for HMAC derivation keys |
| Initialization Vector | Octet String | No, depends on PRF and mode of operation: empty IV is assumed if not provided. |
| Derivation Data | Octet String | Yes, unless the Unique Identifier of a Secret Data |

 

| | | |
|---|---|---|
| | | object is provided |

Cryptographic Parameters identify the Pseudorandom Function (PRF) or the mode of operation of the PRF. For example, if a key is derived using the HASH derivation method, clients are required to provide the hash algorithm inside Cryptographic Parameters. Similarly, if a key is derived using AES in CBC mode, clients are required to provide the Block Cipher Mode. The server will verify that the specified mode matches one of the instances of Cryptographic Parameters set for the corresponding key. If Cryptographic Parameters are omitted, the server will pick the Cryptographic Parameters set with the lowest index for the specified key. If the corresponding key does not have any Cryptographic Parameters attribute, or if no match is found, an error is returned.

If a key is derived using HMAC, the attributes of the derivation key provides enough information about the PRF and Cryptographic Parameters are ignored.

Derivation Data can either be the data to be encrypted, hashed, or HMACed. For NIST SP 800-108 methods, Derivation Data is Label||{0x00}||Context, where the all-zero octet is optional.

Most derivation methods, such as ENCRYPT, require a derivation key and the derivation data to be encrypted. The HASH derivation method requires either a derivation key or derivation data. Derivation data can either be explicitly provided by the client with the Derivation Data field or implicitly by providing the Unique Identifier of a Secret Data object. An error is returned if both are provided.

The PBKDF2 derivation method requires two additional parameters:

| Object | Encoding | Required Field |
|---|---|---|
| Derivation Parameters | Structure | Yes |
| Cryptographic Parameters | Structure | No, depends on the PRF |
| Initialization Vector | Octet String | No, depends on PRF and mode of operation: empty IV is assumed if not provided. |
| Derivation Data | Octet String | Yes, unless the Unique Identifier of a Secret Data object is provided |
| Salt | Octet String | Yes |
| Iteration Count | Integer | Yes |

# 6 Certify

This request is used to obtain a new certificate for a public key. Only a single certificate can be requested at a time. Server support for this operation is optional, as it requires that the key management system have access to a certification authority.

Requests are passed as Octet Strings, which allow multiple certificate request types for X.509 certificates (e.g. PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

The server must copy the Unique Identifier of the certificate returned by this operation into the ID Placeholder variable. The new Certificate object whose Unique Identifier is returned may be obtained by the client via a Get operation in the same batch, using the ID Placeholder mechanism.

As a result of Certify, the Link attribute of the Public Key and of the new Certificate are set to point at each other.

The server must copy the Unique Identifier of the new certificate returned by this operation into the ID Placeholder variable.

 

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

If the information in the Certificate Request conflicts with the attributes specified in the Template-Attribute, then the information in the Certificate Request takes precedence.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | The Unique Identifier of the Public Key being certified. If omitted, the ID Placeholder is substituted by the server |
| Certificate Request Type | Yes | An Enumeration object specifying the type of certificate request |
| Certificate Request | Yes | An Octet String object with the certificate request |
| Template-Attribute | No | Specifies desired object attributes using templates and/or as individual attributes |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the new certificate |
| Template-Attribute | No | A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |

# 7 Re-certify

This request is used to renew an existing certificate with the same key pair. Only a single certificate can be renewed at a time. Server support for this operation is optional, as it requires that the key management system have access to a certification authority.

Requests are passed as Octet Strings, which allow multiple certificate request types for X.509 certificates (e.g. PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

The server must copy the Unique Identifier of the certificate returned by this operation into the ID Placeholder variable.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

If the information in the Certificate Request conflicts with the attributes specified in the Template-Attribute, then the information in the Certificate Request takes precedence.

As the new certificate takes over the name attribute of the existing certificate, Re-certify should only be performed once on a given certificate.

As a result of Re-certify, attributes of the existing certificate are changed similarly to performing a Revoke on that key with a Revocation Reason of Superseded.

 

In addition, the Link attribute of the existing certificate and of the new certificate are set to point at each other. In addition, the Link attribute of the Public Key is changed to point to the new certificate. If *Offset* is set, then the times of the new certificate will be set based on the times of the existing certificate (if such times exist) as follows:

| Attribute in Existing Certificate | Attribute in New Certificate |
|---|---|
| Initial Date ($IT_1$) | Initial Date ($IT_2$) > $IT_1$ |
| Activation Date ($AT_1$) | Activation Date ($AT_2$) = $IT_2$+ *Offset* |
| Deactivation Date ($DT_1$) | Deactivation Date ($DT_1$+($AT_2$- $AT_1$)) |

Attributes that are not copied from the existing certificate and are handled in a specific way are:

| Attribute | Action |
|---|---|
| Initial Date | Set to current time |
| Destroy Date | Not set |
| Revocation Reason | Not set |
| Unique Identifier | New value generated |
| Name | Set to the name(s) of the existing certificate; all name attributes of the existing certificate are removed. |
| State | Set based on attributes |
| Digest | Recomputed from the new certificate value |
| Link | Set to point to the existing certificate as the replaced certificate |
| Last Change Date | Set to current time |

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | The Unique Identifier of the Certificate being renewed. If omitted, the *ID Placeholder* is substituted by the server |
| Certificate Request Type | Yes | An Enumeration object specifying the type of certificate request |
| Certificate Request | Yes | An Octet String object with the certificate request |
| Offset | No | An Interval object indicating the difference between the Initialization Time of the new certificate and the Activation Date of the new certificate |
| Template-Attribute | No | Specifies desired object attributes using templates and/or as individual attributes |

 

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the new certificate |
| Template-Attribute | No | A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here |

# 8 Locate

This operation requests that the server searches for one or more Managed Objects, specified by one or more attributes. All attributes are allowed to be used. However, no attributes specified in the request should contain index values. Attribute Index values will be ignored by the *Locate* operation. The request may also contain a *Maximum Items* field, which specifies the maximum number of objects that the client wishes returned by Locate. If the Maximum Items field is omitted, then the server may return all objects matched, or may impose an internal maximum limit due to resource limitations.

The response may contain Unique Identifiers for multiple Managed Objects, if more than one object satisfies the identification criteria specified in the request. Returned objects must match **all** of the attributes in the request. If no objects match, an empty response payload is returned.

The server returns a list of Unique Identifiers of the found objects, which then must be retrieved using the Get operation, or if the objects are archived, then the Recover and Get operations must be used. The server must copy the Unique Identifier returned by this operation into the ID Placeholder variable. If the Locate operation matches more than one object, and the Maximum Items value is omitted in the request, or is set to a value larger than one, then the server must not set the ID Placeholder value, so that any subsequent operations that are batched with the Locate, and which do not specify a Unique Identifier explicitly will fail. This ensures that these batched operations will be allowed to proceed only if a single object is returned by Locate.

When using the Name or Object Group attributes for identification, wild-cards or regular expressions may be supported by specific key management system implementations. The protocol neither requires nor disallows such use.

The Date attributes (Initial Date, Activation Date, etc) may be used to specify a time or a time range. If a single instance of a given Date attribute is used, such as Activation Date, then objects with the same Activation Date are matching candidate objects. If two instances of the same Date attribute are used (with two different values specifying a range), then objects for which the Activation Date is inside or on the range are matching candidate objects. If a Date attribute is set to its largest possible value, then it is equivalent to an undefined attribute.

When the Cryptographic Usage Mask attribute is specified in the request, candidate objects are matched against this field via an operation which consists of a logical AND of the requested mask with the mask in the candidate object and then a straight comparison of the resulting value with the requested mask. For example, if the request contains a mask value of 1000110001 and a candidate object mask contains 1000010001, the logical AND of the two masks is 1000010001 which is compared against 1000110001 and fails the match. This means that a matching candidate object must have all of the bits set in its mask that are set in the requested mask, but may have additional bits set.

When the Usage Allocation attribute is specified in the request, matching candidate objects must have an Object or Byte Count and Total Objects or Bytes equal or larger than the values specified in the request.

When an attribute defined as a structure is specified, not all of the structure fields must be specified. For instance, for the Link attribute, the Linked Object Identifier value may be specified without the Link Type value, and matching candidate objects must have the Linked Object Identifier as specified, irrespective of their Link Type.

 

The Storage Status Mask field (see Section 9.1.3.3.2 ) is used to indicate whether only on-line objects, or only archived objects, or both on-line and archived objects must be searched. Note that the server may store attributes of archived objects in order to expedite Locate operations searching through archived objects.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Maximum Items | No | An Integer object that indicates the maximum number of object identifiers the server should return |
| Storage Status Mask | No | An Integer object (used as a bit mask) that indicates whether only on-line objects, or only archived objects, or both on-line and archived objects must be searched. If omitted, on-line only is assumed. |
| Attribute | Yes, may be repeated | Specifies an attribute and its value that must match the desired object |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No, May be repeated | The Unique Identifier of the located objects |

# 9 Check

This operation requests that the server checks for the use of a Managed Object according to policy-related values specified in the request. This operation should only be used when placed in a batched set of operations, usually following a Locate, Create, Create Pair, Derive Key, Certify, Re-Certify or Re-Key operation and followed by a Get operation. The Unique Identifier field in the request may be omitted if the operation is in a batched set of operations and follows an operation that sets the ID Placeholder variable.

If the server determines that the client is allowed to use the object specified according to the given policy attributes , the server returns the Unique Identifier of the object. If the server determines that the specified attributes fall outside allowed policy, then the server returns no Unique Identifier, the server invalidates the ID Placeholder value, and the operation returns the set of attributes specified in the request that caused the server policy denial. Only those attributes that the server judged to be out of policy are returned, allowing the client to determine how to proceed. The operation also returns a failure, thus causing any subsequent operations in the batch to be ignored.

The additional objects that may be specified in the request are limited to (note that these objects are not encoded in an Attribute structure as shown in Section 2.1.1 ):

- Usage Limits Byte Count or Usage Limits Object Count (see Section 3.14 )– The request may contain the usage amount that the client deems necessary to complete its needed function. This does not require that any subsequent Get Usage Allocation operations request this amount. It only means that the client is ensuring that the amount specified is available.
- Cryptographic Usage Mask – This is used to specify the cryptographic operations that the client intends to use the object for (see Section 3.12 ). This allows the server to determine if the policy allows this client to perform these operations with the object. Note that this may be a different value from the one specified in a *Locate* operation that precedes this operation. Locate, for example, may specify a Cryptographic Usage Mask requesting a key that can be

 

used for both Encryption and Decryption, but the value in the Check operation may specify that the the client is only using the key for Encryption at this time.

- Lease Time – This specifies a desired lease time (see Section 3.13 ). The client may use this to determine if the server will allow the client to use the object with the specified lease or longer. Including this attribute in the Check operation does not actually cause the server to grant a lease, but only indicates that the requested lease time value will be granted if requested by a subsequent, batched, Obtain Lease operation.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being requested. If omitted, the ID Placeholder is substituted by the server |
| Usage Limits Byte Count | No | Specifies the number of bytes to be protected to be checked against server policy. May only be present if Usage Limits Object Count is not present |
| Usage Limits Object Count | No | Specifies the number of objects to be protected to be checked against server policy. May only be present if Usage Limits Byte Count is not present |
| Cryptographic Usage Mask | No | Specifies the Cryptographic Usage that the client will use the object for |
| Lease Time | No | Specifies a Lease Time value that the Client is asking the server to validate against server policy |

| **Response Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Usage Limits Byte Count | No | Returned by the Server if the Usage Limits value specified in the Request Payload was larger than the value that the server policy would allow. May only be present if Usage Limits Object Count is not present |
| Usage Limits Object Count | No | Returned by the Server if the Usage Limits value specified in the Request Payload was larger than the value that the server policy would allow. May only be present if Usage Limits Byte Count is not present |
| Cryptographic Usage Mask | No | Returned by the Server if the Cryptographic Usage Mask specified in the Request Payload was rejected by the server for policy violation |
| Lease Time | No | Returned by the Server if the Lease Time value in the Request Payload was larger |

 

| | | |
|---|---|---|
| | | than a valid Lease Time that the server would grant |

The encodings of the Usage limits Byte and Object Counts is as shown in Section 3.14 .

# 10 Get

This operation requests that the server returns a Managed Object, which is specified in the request by its Unique Identifier. The Unique Identifier field in the request may be omitted if the *Get* operation is in a batched set of operations and follows an operation that sets the ID Placeholder variable.

Only a single object is returned. Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified. The response contains the Unique Identifier of the object along with the object itself, which may be optionally wrapped using a wrapping key specified in the request.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being requested. If omitted, the ID Placeholder is substituted by the server |
| Key Wrapping Specification | No | Specifies keys and other information for wrapping the returned object. This field may not be specified if the returned object is a Template or Policy Template |

| **Response Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Object Type | Yes | Type of object |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template, Policy Template, Secret Data, or Opaque Object | Yes | The cryptographic object being returned |

# 11 Get Attributes

Return one or more attributes of a Managed Object. The object is specified by its Unique Identifier. The desired attributes are specified by name in the request. If a specified attribute has multiple instances, all instances are returned. If a specified attribute does not exist (i.e. has no value) it must not be present in the returned response. If no requested attributes exist, the response should consist only of the Unique Identifier. Note that the response payload is empty if there are no attribute values to return.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object whose attributes are being requested. If omitted, the ID |

 

| | | |
|---|---|---|
| | | Placeholder is substituted by the server |
| Attribute Name | Yes, May be repeated | Specifies a desired attribute of the object |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Attribute | No, May be repeated | The requested attribute for the object |

# 12 Get Attribute List

Returns a list of the attribute names associated with a specified object. The object is specified by its Unique Identifier.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object whose attribute names are being requested. If omitted, the ID Placeholder is substituted by the server |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Attribute Name | Yes, May be repeated | The requested attribute names for the object |

# 13 Add Attribute

This request adds a new attribute (with possibly multiple values) and sets its value. The request contains the Unique Identifier of the Managed Object which the attribute pertains to, and the name and new value of the attribute. Only non-existing attributes may be set via this operation. Existing attributes must be changed by the Modify Attribute operation. Read-Only attributes may not be added using this operation. No Attribute Index may be specified in the request. The response will return a new Attribute Index if the attribute being added is allowed to have multiple instances. Multiple Add Attribute requests may be included in a single batched request to add multiple attributes.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |

 

| | | |
|---|---|---|
| Unique Identifier | No | The Unique Identifier of the object. If omitted, the ID Placeholder is substituted by the server |
| Attribute | Yes | Specifies the attribute of the object to be added |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Attribute | Yes | The added attribute |

# 14 Modify Attribute

This request modifies the value of an existing attribute. The request contains the Unique Identifier of the Managed Object which the attribute pertains to, and the name, optional index, and new value of the attribute. Only existing attributes may be changed via this operation. New attributes must be added by the Add Attribute operation. Read-Only attributes may not be changed using this operation. If an attribute index is specified, only the specified instance is modified. If the attribute has multiple instances and no index is specified in the request, then the index is assumed to be 0. If the attribute does not support multiple instances, the attribute index must not be specified.

The Attribute returned in the response may have a value different from the one sent in the request, if the server policy so dictates. The value returned is the value set by the server.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | The Unique Identifier of the object. If omitted, the ID Placeholder is substituted by the server |
| Attribute | Yes | Specifies the attribute of the object to be modified |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Attribute | Yes | The modified attribute |

# 15 Delete Attribute

This request deletes an attribute. The request contains the Unique Identifier of the Managed Object which the attribute pertains to, and the name, and optionally the Attribute Index of the attribute. Required attributes and Read-Only attributes may not be deleted by this operation. If no Attribute Index is specified, and the Attribute whose name is specified has multiple instances, the operation is rejected. Note that only a single attribute can be deleted at a time. Multiple delete operations

 

(possible batched) are necessary to delete several attributes. Deleting non-existing attributes will result in an error. Using a non-existing attribute index in a delete operation will also result in an error.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object whose attributes are being updated. If omitted, the ID Placeholder is substituted by the server |
| Attribute Name | Yes | Specifies the name of the attribute of the object to be deleted |
| Attribute Index | No | Specifies the Index of the Attribute |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Attribute | Yes | The deleted attribute |

# 16 Obtain Lease

This request is used to request a new *Lease Time* for a specified cryptographic object. The Lease Time is an interval value that determines when the client's internal cache of information about the object expires and must be renewed. If the returned value of the lease time is zero, then the server is indicating that no lease interval is effective and the client may use the object without any lease time limit. If a client's lease expires, the client must not use the associated cryptographic object until a new lease is obtained. If the server determines that a new lease should not be issued for the specified cryptographic object, then the server should respond to the Obtain Lease request with a Result Status of Failure, and a Result Reason of General Failure.

The response payload for the operation also contains the current value of the Last Changed Date attribute for the object. This may be used by the client to determine if any of the attributes cached by the client need to be refreshed, by comparing this time to the time when the attributes were previously obtained.

The Unique Identifier field in the request may be omitted if the operation is in a batched set of operations and follows an operation that sets the ID Placeholder variable.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object for which the lease is being obtained. If omitted, the *ID Placeholder* is substituted by the server |

| Response Payload | | |
|---|---|---|
| **Object** | **Required** | **Description** |

 

| | Field | |
|---|---|---|
| Unique Identifier | Yes | The Unique Identifier of the object |
| Lease Time | Yes | An interval (in seconds) determining the amount of time that the object can be used until a new lease needs to be obtained |
| Last Changed Date | Yes | The date and time indicating when the latest change was made to the contents or any attribute of the specified object. |

# 17 Get Usage Allocation

This request is used to obtain an allocation from the current Usage Limits values, to allow the client to use the Managed Cryptographic Object for protection purposes. It only applies to Managed Cryptographic Objects that can be used for protection purposes (symmetric keys, private keys and public keys) and is only valid if the Managed Cryptographic Object has a Usage Limits attribute. Usage for process purposes (decryption, verification, etc.) is not limited and cannot be allocated. A Managed Cryptographic Object that has a Usage Limits attribute may not be used by a client for protection purposes unless an allocation has been obtained using this operation. The operation may only be issued during the time that protection is enabled for these objects, i.e. after the Activation Date and before the Protect Stop Date. If the operation is issued for an object that has no Usage Limits attribute, or is not an object that can be used for protection purposes, the server must return a response with a Result Reason of Operation Not Supported.

The fields in the request specify the number of bytes, or number of objects that the client needs to protect. Exactly one of the two count fields must be specified in the request. The corresponding field, containing the number of bytes, or number of objects that may be protected, is returned in the response. If the requested amount is not available, the server may return a smaller amount, or may return 0, indicating that the Managed Object may not be used for protection purposes at this time. The server must assume that the entire allocated amount has been consumed. Server policy may allow the value returned in the response to be different from the value requested. Once the entire allocated amount has been consumed, the client may not continue to use the Managed Cryptographic Object for protection purposes until a new allocation is obtained.

The Unique Identifier field in the request may be omitted if the operation is in a batched set of operations and follows an operation that sets the ID Placeholder variable.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object whose usage allocation is being requested. If omitted, the ID Placeholder is substituted by the server |
| Usage Limits Byte Count | No | The number of bytes to be protected. May only be present if Usage Limits Object Count is not present |
| Usage Limits Object Count | No | The number of objects to be protected. May only be present if Usage Limits Byte Count is not present |

| **Response Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |

 

| | | |
|---|---|---|
| Unique Identifier | Yes | The Unique Identifier of the object |
| Usage Limits Byte Count | No | The number of bytes that may be protected. May only be present if Usage Limits Object Count is not present |
| Usage Limits Object Count | No | The number of objects that may be protected. May only be present if Usage Limits Byte Count is not present |

The encodings of the Usage limits Byte and Object Counts is as shown in Section 3.14 .

# 18 Activate

This request is used to activate a Managed Cryptographic Object. The request may not specify a Template or Policy Template object. The request contains the unique identifier of the Managed Cryptographic Object . The operation can be performed only on an object in the Pre-Active state and has the effect of changing its state to Active and its Activation Date will be set to the current date and time.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being activated. If omitted, the ID Placeholder is substituted by the server |

| **Response Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |

# 19 Revoke

This request is used to revoke a Managed Cryptographic Object. The request may not specify a Template or Policy Template object. The request contains the unique identifier of the Managed Cryptographic Object and a reason for the revocation, e.g. “compromised”, “no longer used”, etc. Special authentication and authorization is required to issue this request (see Usage Guide). Only the object creator or an authorized security officer should be allowed to issue this request. The operation will have one of two effects. If the revocation reason is “compromised”, then the object will be placed into the “compromised” state, and the Compromise Date attribute will be set to the current date and time. Otherwise, the object will be placed into the “deactivated” state, and the Deactivation Date attribute will be set to the current date and time.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being revoked. If omitted, the ID Placeholder is substituted by the server |

 

| | | |
|---|---|---|
| Revocation Reason | Yes | Specifies the reason for revocation |
| Compromise Occurrence Date | No | Only specified, and required, if the Revocation Reason is 'compromised' |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |

## 20 Destroy

This request is used to indicate to the server that the key material for the specified Managed Cryptographic Object should be destroyed. The meta-data for the key material may be retained by the server. This is used for example, to ensure that copies of an expired or revoked private signing key are no longer available. Special authentication and authorization is required to issue this request (see Usage Guide). Only the object creator or an authorized security officer should be allowed to issue this request. If the Unique Identifier specifies a Template or Policy Template object, then the object itself, including all meta-data may be destroyed.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being destroyed. If omitted, the ID Placeholder is substituted by the server |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |

## 21 Archive

This request is used to specify that a Managed Object is now permitted to be placed in archival storage. The actual time when the object is placed in archival storage and the location of the archive or level of archive hierarchy is determined by the policies within the key management system, and is not specified by the client. The request contains the unique identifier of the object. Special authentication and authorization is required to issue this request (see Usage Guide). Only the object creator or an authorized security officer should be allowed to issue this request. This request may be considered only a “hint” to the key management system, which may or may not choose to act upon this request.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being archived. If omitted, the ID Placeholder is substituted by the server |

 

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |

## 22 Recover

This request is used to obtain access to a Managed Object that has been placed in archival storage. Due to the fact that the object is located in archival storage, this request may require asynchronous polling to obtain the response. Once the response is received, the object is now on-line, and may be obtained via a normal Get operation, for instance. Special authentication and authorization is required to issue this request (see Usage Guide).

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | No | Determines the object being recovered. If omitted, the ID Placeholder is substituted by the server |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |

## 23 Validate

This requests that the server validate a certificate chain, and return information on its validity. Only a single certificate chain may be included in each request. Support for this operation at the server is optional.

The request may contain a list of certificate objects, and/or a list of Unique Identifiers which identify Managed Certificate objects. The two lists must together comprise a certificate chain to be validated. The request may also optionally contain a date for which the certificate chain must be valid.

The validation method or policy by which validation will be conducted is a decision of the server and is outside of the scope of this protocol. Likewise, the order in which the supplied certificate chain is validated and the specification of trust anchors used to terminate validation are also controlled by the server.

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Certificate | No, May be repeated | One or more Certificates |
| Unique Identifier | No, May be repeated | One or more Unique Identifiers of Certificate Objects |

 

| | | |
|---|---|---|
| Validity Date | No | A Date-Time object indicating when the certificate chain must be valid |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Validity Indicator | Yes | An Enumeration object indicating whether the certificate chain is valid, invalid, or unknown |

# 24 Query

This request is used by the client to interrogate the server to determine its capabilities and/or protocol mechanisms. It is recommended that the *Query* operation, used to interrogate server features and functions, be invocable by unauthenticated clients. The *Query Function* field in the request may contain one of the following items:

- Query Operations
- Query Objects
- Query Server Information

One, two, or all three of the above functions may be specified.

The *Operation* fields in the response contain Operation enumerated values, which should list the optionally supported operations that the server supports. These fields should only be returned in the response if the request contains a Query Operations value in the Query Function field. The optional operations are:

- Validate
- Certify
- Re-Certify
- Notify
- Put

The *Object Type* fields in the response contain Object Type enumerated values, which should list the object types that the server supports. These fields should only be returned in the response if the request contains a *Query Objects* value in the Query Function field. The object types (any of which are optional) are:

- Certificate
- Symmetric Key
- Public Key
- Private Key
- Split Key
- Template
- Policy Template
- Secret Data
- Opaque Object

The *Server Information* field in the response is a structure containing vendor specific fields and/or substructures. This field should only be returned in the response if the request contains a *Query Server Information* value in the Query Function field.

 

Note that the response payload is empty if there are no values to return.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Query Function | Yes, May be Repeated | Determines the information being queried |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Operation | No, May be repeated | Specifies an Operation that is supported by the server. Only optional operations should be listed |
| Object Type | No, May be repeated | Specifies a Managed Object Type that is supported by the server |
| Vendor Identification | No | Must be returned if Query Server Information is requested. The Vendor Identification must be a text string that uniquely identifies the vendor |
| Server Information | No | Contains vendor-specific information that may be of interest to the client |

# 25 Cancel

This request is used to cancel an outstanding asynchronous operation. The correlation value (see Section 6.8 ) of the original operation must be specified in the request. The server must respond with a *Cancellation Result*, which contains one of the following values:

- *Canceled* – The cancel operation succeeded in canceling the pending operation.
- *Unable To Cancel* – The cancel operation is unable to cancel the pending operation.
- *Completed* – The pending operation completed successfully before the cancellation operation was able to cancel it.
- *Failed* – The pending operation completed with a failure before the cancellation operation was able to cancel it.
- *Unavailable* – The specified correlation value did not match any recently pending or completed asynchronous operations.

The response to this operation cannot be asynchronous.

| Request Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Asynchronous Correlation Value | Yes | Specifies the request being canceled |

| Response Payload | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Asynchronous Correlation Value | Yes | Specifies the request |

 

| | | |
|---|---|---|
| Cancellation Result | Yes | Enumeration indicating result of cancellation |

## 26 Poll

This request is used to poll the server in order to obtain the status of an outstanding asynchronous operation. The correlation value (see Section 6.8 ) of the original operation must be specified in the request. The response to this operation cannot be asynchronous.

| **Request Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Asynchronous Correlation Value. | Yes | Specifies the request being polled |

The server must reply with one of two responses:

- A response containing no payload and a Result Status of Pending, if the operation has not completed
- A response containing the appropriate payload for the operation, if the operation has completed. This response must be identical to the response that would have been sent if the operation had completed synchronously.

# 5 Server-to-Client Operations

Server-to-client operations are used by servers to send information or Managed Cryptographic Objects to clients outside of the normal client-server request-response mechanism. These operations are used to "push" Managed Cryptographic Objects directly to clients without a specific request from the client.

## 1 Notify

This operation is used to notify a client of events. This operation is only ever sent by a server to a client outside the normal client request/response protocol, using information known to the server via unspecified configuration or administrative mechanisms. It contains the Unique Identifier of the object to which the notification applies, and a list of the attributes whose changed values have triggered the notification. The message is sent as a normal Request message, except that the Maximum Response Size, Asynchronous Indicator, Batch Error Continuation Option, and Batch Order Option fields are not allowed. The client must send a response in the form of a Response Message containing no payload, unless both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the client cannot respond. Server and Client support for this message is optional.

| **Message Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description p** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Attribute | Yes, May be repeated | The attributes which have changed. This includes at least the Last Changed Date attribute |

## 2 Put

This operation is used to "push" Managed Cryptographic Objects to clients. This operation is only ever sent by a server to a client outside the normal client request/response protocol, using information known to the server via unspecified configuration or administrative mechanisms. It contains the Unique Identifier of the object which is being sent, and the object itself. The

 

message is sent as a normal Request message, except that the Maximum Response Size, Asynchronous Indicator, Batch Error Continuation Option, and Batch Order Option fields are not allowed. The client must send a response in the form of a Response Message containing no payload, unless both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the client cannot respond. Server and client support for this message is optional.

The *Put Function* field indicates whether the object being "pushed" is a new object, or a replacement for an object already known to the client. For example, when pushing a certificate to replace one that is about to expire, the Put Function field would be set to indicate replacement, and the Unique Identifier of the expiring certificate would be placed in the *Replaced Unique Identifier* field. The Put Function may contain one of the following values:

- *New* – which indicate that the object is not a replacement for another object
- *Replace* – which indicates that the object is a replacement for another object, and that the Replaced Unique Identifier field is present, and contains the identification of the replaced object.

The Attribute field contains one or more attributes that the server wishes to be pushed along with the object. In particular, the server may include policy attributes with the object to specify how the object is to be used by the client. The server may include a Lease Time attribute which grants a lease to the client.

If the Managed Object is a wrapped key, the key wrapping specification must be exchanged prior to the transfer via out-of-band mechanisms.

| **Message Payload** | | |
|---|---|---|
| **Object** | **Required Field** | **Description** |
| Unique Identifier | Yes | The Unique Identifier of the object |
| Put Function | Yes | Indicates function for Put message |
| Replaced Unique Identifier | No | Unique Identifier of replaced object. Must be present if the *Put Function* is *Replace* |
| Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template, Policy Template, Secret Data, or Opaque Object | Yes | The object being sent to the client |
| Attribute | No, May be repeated | The additional attributes that the server wishes to push with the object |

# 6 Message Contents

Messages in the protocol consist of a message header and one or more batch items which contain optional message payloads, and optional message extensions. The message headers contain fields whose presence is determined by the protocol features used, e.g. asynchronous responses. The field contents are also determined by whether the message is a request or a response. The message payload is determined by the specific operation being requested or replied to.

The message headers are structures which contain some of the following objects.

## 1 Protocol Version

This field contains the version number of the protocol, ensuring that the protocol is fully understood by both communicating parties. The version number is specified in two parts, major and minor. Servers and clients must support backward compatibility with versions of the protocol with the same major version but different minor versions. Support for backward compatibility with different major versions is optional.

| **Object** | **Encoding** | **Required Field** |
|---|---|---|

 

| | | |
|---|---|---|
| Protocol Version | Structure | Yes |
| Protocol Version Major | Integer | Yes |
| Protocol Version Minor | Integer | Yes |

## 2 Operation

This field indicates the operation being requested or the operation for which the response is being returned. The operations are defined in Sections 4 and 5 .

| Object | Encoding | Required Field |
|---|---|---|
| Operation | Enumeration | Yes |

## 3 Maximum Response Size

This field is optionally contained in a request message, and is used to indicate the maximum size of a response that the requester can handle. It need only be sent in requests that may return large replies.

| Object | Encoding | Required Field |
|---|---|---|
| Maximum Response Size | Integer | No |

## 4 Unique Message ID

This field is optionally contained in a request, and is used as for correlation between requests and responses. If a request has a *Unique Message ID*, then responses to that request must have the same Unique Message ID.

| Object | Encoding | Required Field |
|---|---|---|
| Unique Message ID | Octet String | No |

## 5 Time Stamp

This field is optionally contained in a request and required in a response, and is used for time stamping and may be used to enforce reasonable time usage at a client, e.g. a server may choose to reject a request if a client's time stamp contains a value that is too far off the known correct time. It may also be used by a client, which has no real-time clock but only a countdown timer, to obtain useful “seconds from now” values from all of the Date attributes, by performing a subtraction.

| Object | Encoding | Required Field |
|---|---|---|
| Time Stamp | Date-Time | No |

## 6 Authentication

This is used to authenticate the requester. It is an optional information item, depending on the type of request being issued and on server policies. Servers may require authentication on no requests, a subset of the operations, or all requests, depending on policy. It is recommended that the Query operation, used to interrogate server features and functions, not require authentication.

 

The authentication mechanisms are described and discussed in Section 8 .

| Object | Encoding | Required Field |
|---|---|---|
| Authentication | Structure | No |
| Credential | Structure | Yes |

The Credential structure is defined in Section 2.1.2 .

# 7 Asynchronous Indicator

This boolean flag indicates whether the client can accept an asynchronous response. It must have the boolean value True if the client can handle asynchronous responses, and the value False otherwise. If not present in a request, False is assumed. If a client indicates that it can't handle asynchronous responses (flag is set to False) and the server is not capable to process the request synchronously, the server must reject the request with a failure status.

| Object | Encoding | Required Field |
|---|---|---|
| Asynchronous Indicator | Boolean | No |

# 8 Asynchronous Correlation Value

This is returned in the immediate response to an operation that will require asynchronous polling. It is a server generated correlation value that must be specified in any subsequent Poll, or Cancel operations that pertain to the original operation.

| Object | Encoding | Required Field |
|---|---|---|
| Asynchronous Correlation Value | Octet String | No |

# 9 Result Status

This indicates the success or failure of the request. The following values may be set in this field:

- *Success* – The requested operation completed successfully.
- *Pending* – The requested operation is in progress and the actual result must be obtained via asynchronous polling. The asynchronous correlation value must be used for the subsequent polling of the result status.
- *Undone* – The requested operation was performed but had to be undone (due to a failure in a batch for which the Error Continuation Option was set to Undo)
- *Failure* – The requested operation failed.

| Object | Encoding | Required Field |
|---|---|---|
| Result Status | Enumeration | Yes |

# 10 Result Reason

This field indicates a reason for failure or a modifier for a partially successful operation and must be present in responses that return a Result Status of Failure. It is optional in any response that returns a Result Status of Success. The following defined values may be set in this field:

- *Item Not Found* – A requested object was not found or did not exist.
- *Response too large* – The response to a request would have exceeded the *Maximum Response Size* in the request.

 

- *Authentication not successful* – The authentication in the request did not pass validation, or there was no authentication in the request when there should have been.
- *Invalid Message* – The request message was not understood by the server.
- *Operation Not Supported* – The operation requested by the request message was not supported by the server.
- *Missing Data* – The operation required additional optional information in the request, which was not present.
- *Invalid Field* – Some data item in the request had an invalid value.
- *Feature not supported* – An optional feature specified in the request was not supported.
- *Operation canceled by requester* – The operation was asynchronous and the operation was canceled by the Cancel operation before it completed successfully.
- *Cryptographic failure* – The operation failed due to a cryptographic error.
- *Illegal Operation* – The client requested an operation that could not be performed with the specified parameters.
- *Permission Denied* – The client did not have permission to perform the requested operation.
- *Object archived* – The object should be first recovered from the archive.
- *General Failure* – The request failed for a reason other than the defined reasons above.

| Object | Encoding | Required Field |
|---|---|---|
| Result Reason | Enumeration | Yes |

# 11 Result Message

This field may optionally be returned in a response. It contains a more descriptive error message, which may be used by the client to display to an end user or for logging/auditing purposes.

| Object | Encoding | Required Field |
|---|---|---|
| Result Message | Text String | No |

# 12 Batch Order Option

A Boolean value used in requests where the Batch Count is greater than 1. If true then batched operations must be executed in the order in which they appear within the request. If false, the server may choose to execute the batched operations in any order. If not specified, false is assumed (i.e. no implied ordering). Server support for this feature is optional, but if the server does not support the feature, and a request is received with the flag true, the entire request must be rejected.

| Object | Encoding | Required Field |
|---|---|---|
| Batch Order Option | Boolean | No |

# 13 Batch Error Continuation Option

Batched operation partial failure continuation option. This option should only be present if the Batch Count is greater than 1. This option may have one of three values:

- *Undo* – If any operation in the request fails, the server must undo all the previous operations.
- *Stop* – If an operation fails, the server must not continue processing later operations in the request. Completed operations will be left intact.

 

- *Continue* – Return an error for the failed operation and continue processing later operations in the request.

If not specified, Stop is assumed.

Server support for this feature is optional, but if the server does not support the feature, and a request is received containing the *Batch Error Continuation* option, the entire request must be rejected.

| Object | Encoding | Required Field |
|---|---|---|
| Batch Error Continuation Option | Enumeration | No |

## 14 Batch Count

This field is required. It contains the number of Batch Items in a message. If only a single operation is being requested, the batch count must be set to 1. The Message Payload, which follows the Message Header, will contain one or more batch items.

| Object | Encoding | Required Field |
|---|---|---|
| Batch Count | Integer | Yes |

## 15 Batch Item

This field is required. It consists of a structure that holds the individual requests or responses in a batch. The contents of the batch items is described in Sections 7.2 and 7.3 .

| Object | Encoding | Required Field |
|---|---|---|
| Batch Item | Structure | No |

## 16 Message Extension

The *Message Extension* is an optional structure which may be appended to any Batch Item. It is used to extend protocol messages for the purpose of adding vendor specified extensions. The Message Extension is a structure containing a Vendor Identification, Criticality Indicator, and vendor-specific extensions. The *Vendor Identification* must be a text string that uniquely identifies the vendor, allowing a client to determine if the extension can be parsed and understood. If a client or server receives a protocol message containing a message extension that it does not understand, its actions depend on the *Criticality Indicator*. If the indicator is True (Critical), and the receiver does not understand the extension, the receiver must reject the entire message. If the indicator is False (Non-Critical), and the receiver does not understand the extension, the receiver may process the rest of the message as if the extension were not present.

| Object | Encoding | Required Field |
|---|---|---|
| Message Extension | Structure | No |
| Vendor Identification | Text String | Yes |
| Criticality Indicator | Boolean | Yes |
| Vendor Extension | Structure | Yes |

# 6 Message Format

Messages contain the following objects and fields. All fields must appear in the order specified.

 

# 1 Message Structure

| Object | Encoding | Required Field |
|---|---|---|
| Request Message | Structure | Yes |
| Request Header | Structure | Yes |
| Batch Item | Structure | Yes, May be repeated |

| Object | Encoding | Required Field |
|---|---|---|
| Response Message | Structure | Yes |
| Response Header | Structure | Yes |
| Batch Item | Structure | Yes, May be repeated |

# 2 Synchronous Operations

| Synchronous Request Header | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Request Header | Yes | Structure |
| Protocol Version | Yes | |
| Maximum Response Size | No | |
| Authentication | No | |
| Batch Error Continuation Option | No | If omitted, Stop is assumed |
| Batch Order Option | No | If omitted, False is assumed |
| Time Stamp | No | |
| Batch Count | Yes | |

| Synchronous Request Batch Item | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Batch Item | Yes | Structure |
| Operation | Yes | |
| Unique Message ID | No | Required if *Batch Count* > 1 |
| Request Payload | Yes | Structure, contents depend on the Operation |
| Message Extension | No | |

| Synchronous Response Header | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Response Header | Yes | Structure |

 

| | | |
|---|---|---|
| Protocol Version | Yes | |
| Time Stamp | Yes | |
| Batch Count | Yes | |

| Synchronous Response Batch Item | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Batch Item | Yes | Structure |
| Operation | Yes, if not a failure | |
| Unique Message ID | No | Required if *Batch Count* > 1 |
| Result Status | Yes | |
| Result Reason | No | Only present, if Result Status is not *Success* |
| Result Message | No | Only present, if Result Status is not *Success* |
| Response Payload | Yes, if not a failure | Structure, contents depend on the Operation |
| Message Extension | No | |

# 3 Asynchronous Operations

If the client is capable of accepting asynchronous responses, it may set the *Asynchronous Indicator* in the header of a batched request. The batched responses may contain a mixture of synchronous and asynchronous responses.

| Asynchronous Request Header | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Request Header | Yes | Structure |
| Protocol Version | Yes | |
| Maximum Response Size | No | |
| Asynchronous Indicator | Yes | Must be set to True |
| Authentication | No | |
| Batch Error Continuation Option | No | If omitted, Stop is assumed |
| Batch Order Option | No | If omitted, False is assumed |
| Time Stamp | No | |
| Batch Count | Yes | |

| Asynchronous Request Batch Item | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Batch Item | Yes | Structure |

 

| | | |
|---|---|---|
| Operation | Yes | |
| Unique Message ID | No | Required if *Batch Count* > 1 |
| Request Payload | Yes | Structure, contents depend on the Operation |
| Message Extension | No | |

| **Asynchronous Response Header** | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Response Header | Yes | Structure |
| Protocol Version | Yes | |
| Time Stamp | Yes | |
| Batch Count | Yes | |

| **Asynchronous Response Batch Item** | | |
|---|---|---|
| **Object or Field** | **Required in Message** | **Comment** |
| Batch Item | Yes | Structure |
| Operation | Yes, if not a failure | |
| Unique Message ID | No | Required if *Batch Count* > 1 |
| Result Status | Yes | |
| Result Reason | No | Only present, if Result Status is not *Pending* or *Success* |
| Result Message | No | Only present, if Result Status is not *Pending* or *Success* |
| Asynchronous Correlation Value | Yes | Only present, if Result Status is *Pending* |
| Response Payload | Yes, if not a failure | Structure, contents depend on the Operation |
| Message Extension | No | |

# 7 Authentication

The mechanisms used to authenticate the client to the server and the server to the client are not part of the message definitions, and are external to the protocol. The *Authentication* field contained in Request Headers is used to identify the client and to provide linkage between this identification and the external authentication mechanism.

The Usage Guide describes authentication profiles appropriate to this protocol as well as the relationship of those mechanisms to the credentials optionally included in the Authentication field. The authentication profiles described are:

- SSL/TLS authentication. If the transport protocol uses a normal TCP stream, then that stream should use an SSL/TLS encryption layer and the client and server authentication features must be enabled. The Credential object contained in the Authentication field in all request messages will contain the client's certificate. The server should use this certificate to identify the client for policy enforcement purposes, and should verify that this certificate matches the one used for SSL/TLS authentication.

 

- HTTPS authentication. If the transport protocol is HTTP over TCP, then the HTTPS protocol should be used, and the client and server authentication features enabled. The contents and use of the *Credential* object are the same as in the normal TCP example above.

All server implementations should, at least, support the SSL/TLS and HTTPS profiles described in the Usage Guide.

Other mechanisms, such as Kerberos, are potentially usable, with the identity established in the mechanism, such as the Kerberos token, expressed as the Credential object. Profiles for these mechanisms currently are not described in the Usage Guide.

# 8 Message Encoding

To support different transport protocols and different client capabilities, a number of message-encoding mechanisms are supported.

## 1 TTLV Encoding

In order to minimize the resource impact on potentially low-function clients, one encoding mechanism to be used for protocol messages is a simplified TTLV (Tag, Type, Length, Value) scheme.

The scheme is designed to minimize the CPU cycle and memory requirements of clients that must encode or decode protocol messages. Minimizing bandwidth over the transport mechanism is considered to be of lesser importance.

### 1 TTLV Encoding Fields

Every Data object encoded by the TTLV scheme consists of 4 items, in order:

#### *1 Item Tag*

An Item Tag is a 32-bit binary unsigned integer, transmitted big endian, which contains a number that designates the specific Protocol Field or Object that the TTLV object represents. To ease debugging, and to ensure that malformed messages are detected more easily, all tags contain the value 42 in hex as the high order (first) byte. Tags defined by this specification contain hex 00 in the second byte. Extensions, which are permitted, but not defined in this specification, contain the value 01 hex in the second byte. A list of defined Item Tags is in Section 9.1.3.1 .

#### *2 Item Type*

An Item Type is a byte containing a coded value which indicates the data type of the data object. The allowed values are:

| **Data Type** | **Coded Value in Hex** |
|---|---|
| Integer | 01 |
| Long Integer | 02 |
| Big Integer | 03 |
| Enumeration | 04 |
| Boolean | 05 |
| Text String | 06 |
| Octet String | 07 |

 

| | |
|---|---|
| Date-Time | 08 |
| Interval | 09 |
| Structure | 80 |

### *3 Item Length*

An Item Length is a 32-bit binary integer, transmitted big-endian, containing the number of bytes in the Item Value.

Allowed values are:

| **Data Type** | **Length** |
|---|---|
| Integer | 4 |
| Long Integer | 8 |
| Big Integer | Varies |
| Enumeration | 4 |
| Boolean | 1 |
| Text String | Varies |
| Octet String | Varies |
| Date-Time | 8 |
| Interval | 4 |
| Structure | Varies |

If the Item Type is a Big Integer, Text String or Octet String, then the Item Length must be the number of bytes in the string. Strings must not be null-terminated. If the Item Type is a structure, then the Item Length is the total length of all of the sub-items contained in the structure.

### *4 Item Value*

The item value is a sequence of bytes containing the value of the data item, depending on the type:

- Integers are encoded as 4-byte long (32 bit) binary signed numbers in 2's complement notation, transmitted big-endian.
- Long Integers are encoded as 8-byte long (64 bit) binary signed numbers in 2's complement notation, transmitted big-endian.
- Big Integers are encoded as a sequence of 8-bit bytes, in 2's complement notation, transmitted big-endian.
- Enumerations are encoded as 4-byte long (32 bit) binary unsigned numbers transmitted big-endian.

 

- Booleans are encoded as a byte, which must either contain the binary value 00000000, indicating the boolean value *False,* or the binary value 00000001, transmitted big-endian, indicating the boolean value *True.* Values other than 00000000 or 00000001 are to be interpreted as *True*, but are deprecated.
- Text Strings are sequences of bytes encoding character values according to the UTF-8 encoding standard. There must be no null-termination at the end of such strings.
- Octet Strings are sequences of bytes containing individual unspecified 8 bit binary values.
- Date-Time values are encoded as 8-byte long (64 bit) binary signed numbers, transmitted big-endian. They are POSIX Time values (described in IEEE Standard 1003.1) extended to a 64 bit value to eliminate the "Year 2038 problem". The value is expressed as the number of seconds from a time epoch, which is 00:00:00 GMT January 1st, 1970. This value has a resolution of 1 second. All Date-Time values are expressed as UTC values.
- Intervals are encoded as 4-byte long (32 bit) binary unsigned numbers, transmitted big-endian. They have a resolution of 1 second.
- Structure Values are encoded as the concatenated encodings of the elements of the structure. All structures defined in this specification must have all of their fields encoded in the order in which they appear in their respective structure descriptions.

## 2 Examples

These examples are assumed to be encoding a Protocol Object whose tag is 42000020. The examples are shown as a sequence of bytes in hexadecimal notation:

- An Integer containing the decimal value 8:

  42 00 00 20 | 01 | 00 00 00 04 | 00 00 00 08
- A Long Integer containing the decimal value 123456789000000000:

  42 00 00 20 | 02 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00
- A Big Integer containing the decimal value 1234567890000000000000000000:

  42 00 00 20 | 03 | 00 00 00 0C | 03 FD 35 EB 6B C2 DF 46 18 08 00 00
- An Enumeration with value 255:

  42 00 00 20 | 04 | 00 00 00 04 | 00 00 00 FF
- A Boolean with the value *True*:

  42 00 00 20 | 05 | 00 00 00 01 | 01
- A Text String:

  42 00 00 20 | 06 | 00 00 00 0B | 48 65 6C 6C 6F 20 57 6F 72 6C 64
- An Octet String:

  42 00 00 20 | 07 | 00 00 00 03 | 01 02 03
- A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:

  42 00 00 20 | 08 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8
- An Interval, containing the value for 10 days:

  42 00 00 20 | 09 | 00 00 00 04 | 00 0D 2F 00
- A Structure containing an Enumeration, value 254, followed by an Integer, value 255, having tags 42000004 and 42000005 respectively:

  42 00 00 20 | 80 | 00 00 00 1A | 42 00 00 04 | 04 | 00 00 00 04 | 00 00 00 FE | 42 00 00 05 | 01 | 00 00 00 04 | 00 00 00 FF

 

# 3 Defined Values

This section specifies the values that are defined by this specification. In all cases where an extension mechanism is allowed, this extension mechanism may only be used for communication between parties that have pre-agreed understanding of the specific extensions.

 

## *1 Tags*

The following table defines the tag values for the objects and primitive data values for the protocol messages.

| Tag | |
|---|---|
| **Object** | **Tag Value** |
| Activation Date | 42000001 |
| Application Identifier | 42000002 |
| Application Name Space | 42000003 |
| Application Specific Identification | 42000004 |
| Archive Date | 42000005 |
| Asynchronous Correlation Value | 42000006 |
| Asynchronous Indicator | 42000007 |
| Attribute | 42000008 |
| Attribute Index | 42000009 |
| Attribute Name | 4200000A |
| Attribute Value | 4200000B |
| Authentication | 4200000C |
| Batch Count | 4200000D |
| Batch Error Continuation Option | 4200000E |
| Batch Item | 4200000F |
| Batch Order Option | 42000010 |
| Block Cipher Mode | 42000011 |
| Cancellation Result | 42000012 |
| Certificate | 42000013 |
| Certificate Issuer | 42000014 |
| Certificate Request | 42000015 |
| Certificate Request Type | 42000016 |
| Certificate Subject | 42000017 |
| Certificate Subject Alternative Name | 42000018 |
| Certificate Subject Distinguished Name | 42000019 |
| Certificate Type | 4200001A |
| Certificate Value | 4200001B |
| Common Template-Attribute | 4200001C |
| Compromise Date | 4200001D |
| Compromise Occurrence Date | 4200001E |

 

| | |
|---|---|
| Contact Information | 4200001F |
| Credential | 42000020 |
| Credential Type | 42000021 |
| Credential Value | 42000022 |
| Criticality Indicator | 42000023 |
| CRT Coefficient | 42000024 |
| Cryptographic Algorithm | 42000025 |
| Cryptographic Length | 42000026 |
| Cryptographic Parameters | 42000027 |
| Cryptographic Usage Mask | 42000028 |
| Custom Attribute | 42000029 |
| D | 4200002A |
| Deactivation Date | 4200002B |
| Derivation Data | 4200002C |
| Derivation Method | 4200002D |
| Derivation Parameters | 4200002E |
| Destroy Date | 4200002F |
| Digest | 42000030 |
| Digest Value | 42000031 |
| Encryption Key Information | 42000032 |
| G | 42000033 |
| Hashing Algorithm | 42000034 |
| Initial Date | 42000035 |
| Initialization Vector | 42000036 |
| Issuer | 42000037 |
| Iteration Count | 42000038 |
| IV/Counter/Nonce | 42000039 |
| J | 4200003A |
| Key | 4200003B |
| Key Block | 4200003C |
| Key Material | 4200003D |
| Key Part Identifier | 4200003E |
| Key Value | 4200003F |
| Key Value Type | 42000040 |
| Key Wrapping Data | 42000041 |
| Key Wrapping Specification | 42000042 |
| Last Changed Date | 42000043 |
| Lease Time | 42000044 |

 

| | |
|---|---|
| Link | 42000045 |
| Link Type | 42000046 |
| Linked Object Identifier | 42000047 |
| MAC/Signature | 42000048 |
| MAC/Signature Key Information | 42000049 |
| Maximum Items | 4200004A |
| Maximum Response Size | 4200004B |
| Message Extension | 4200004C |
| Modulus | 4200004D |
| Name | 4200004E |
| Name Type | 4200004F |
| Name Value | 42000050 |
| Object Group | 42000051 |
| Object Type | 42000052 |
| Offset | 42000053 |
| Opaque Data Type | 42000054 |
| Opaque Data Value | 42000055 |
| Opaque Object | 42000056 |
| Operation | 42000057 |
| Operation Policy Name | 42000058 |
| P | 42000059 |
| Padding Method | 4200005A |
| Policy Template | 4200005B |
| Prime Exponent P | 4200005C |
| Prime Exponent Q | 4200005D |
| Prime Field Size | 4200005E |
| Private Exponent | 4200005F |
| Private Key | 42000060 |
| Private Key Template-Attribute | 42000061 |
| Private Key Unique Identifier | 42000062 |
| Process Start Date | 42000063 |
| Protect Stop Date | 42000064 |
| Protocol Version | 42000065 |
| Protocol Version Major | 42000066 |
| Protocol Version Minor | 42000067 |
| Public Exponent | 42000068 |
| Public Key | 42000069 |
| Public Key Template-Attribute | 4200006A |

 

| | |
|---|---|
| Public Key Unique Identifier | 4200006B |
| Put Function | 4200006C |
| Q | 4200006D |
| Q String | 4200006E |
| Query Function | 4200006F |
| Recommended Curve | 42000070 |
| Replaced Unique Identifier | 42000071 |
| Request Header | 42000072 |
| Request Message | 42000073 |
| Request Payload | 42000074 |
| Response Header | 42000075 |
| Response Message | 42000076 |
| Response Payload | 42000077 |
| Result Message | 42000078 |
| Result Reason | 42000079 |
| Result Status | 4200007A |
| Revocation Message | 4200007B |
| Revocation Reason | 4200007C |
| Revocation Reason Code | 4200007D |
| Role Type | 4200007E |
| Salt | 4200007F |
| Secret Data | 42000080 |
| Secret Data Type | 42000081 |
| Serial Number | 42000082 |
| Server Information | 42000083 |
| Split Key | 42000084 |
| Split Key Method | 42000085 |
| Split Key Parts | 42000086 |
| Split Key Threshold | 42000087 |
| State | 42000088 |
| Storage Status Mask | 42000089 |
| Symmetric Key | 4200008A |
| Template | 4200008B |
| Template Name | 4200008C |
| Template-Attribute | 4200008D |
| Time Stamp | 4200008E |
| Unique Identifier | 4200008F |
| Unique Message ID | 42000090 |

 

| | |
|---|---|
| Usage Limits | 42000091 |
| Usage Limits Byte Count | 42000092 |
| Usage Limits Object Count | 42000093 |
| Usage Limits Total Bytes | 42000094 |
| Usage Limits Total Objects | 42000095 |
| Validity Date | 42000096 |
| Validity Indicator | 42000097 |
| Vendor Extension | 42000098 |
| Vendor Identification | 42000099 |
| Wrapping Method | 4200009A |
| X | 4200009B |
| Y | 4200009C |
| Extensions | 4201XXXX |

 

## *2 Enumerations*

The following tables define the values for enumerated lists.

### 1 Credential Type Enumeration

| Credential Type | |
|---|---|
| **Name** | **Value** |
| Username & Password | 00000001 |
| Token | 00000002 |
| Biometric Measurement | 00000003 |
| Certificate | 00000004 |
| Extensions | 8XXXXXXX |

### 2 Key Value Type Enumeration

| Key Value Type | |
|---|---|
| **Name** | **Value** |
| Raw | 00000001 |
| Opaque | 00000002 |
| PKCS#1 | 00000003 |
| PKCS#8 | 00000004 |
| Transparent Symmetric Key | 00000005 |
| Transparent DSA Private Key | 00000006 |
| Transparent DSA Public Key | 00000007 |
| Transparent RSA Private Key | 00000008 |
| Transparent RSA Public Key | 00000009 |
| Transparent DH Private Key | 0000000A |
| Transparent DH Public Key | 0000000B |
| Transparent ECDSA Private Key | 0000000C |
| Transparent ECDSA Public Key | 0000000D |
| Transparent ECDH Private Key | 0000000E |
| Transparent ECDH Public Key | 0000000F |
| Extensions | 8XXXXXXX |

### 3 Wrapping Method Enumeration

| Wrapping Method | |
|---|---|
| **Name** | **Value** |

 

| | |
|---|---|
| Encrypt | 00000001 |
| MAC/sign | 00000002 |
| Encrypt then MAC/sign | 00000003 |
| MAC/sign then encrypt | 00000004 |
| TR-31 | 00000005 |
| Vendor specific | 00000006 |
| Extensions | 8XXXXXXX |

## 4 Recommended Curves for ECDSA and ECDH

| **Recommended Curve Enumeration** | |
|---|---|
| **Name** | **Value** |
| P-192 | 00000001 |
| K-163 | 00000002 |
| B-163 | 00000003 |
| P-224 | 00000004 |
| K-233 | 00000005 |
| B-233 | 00000006 |
| P-256 | 00000007 |
| K-283 | 00000008 |
| B-283 | 00000009 |
| P-384 | 0000000A |
| K-409 | 0000000B |
| B-409 | 0000000C |
| P-521 | 0000000D |
| K-571 | 0000000E |
| B-571 | 0000000F |
| Extensions | 8XXXXXXX |

## 5 Certificate Type Enumeration

| **Certificate Type** | |
|---|---|
| **Name** | **Value** |
| X.509 | 00000001 |
| PGP | 00000002 |
| Extensions | 8XXXXXXX |

 

## 6 Split Key Method Enumeration

| Split Key Method | |
|---|---|
| **Name** | **Value** |
| XOR | 00000001 |
| Polynomial Sharing GF($2^{16}$) | 00000002 |
| Polynomial Sharing Prime Field | 00000003 |
| Extensions | 8XXXXXXX |

## 7 Secret Data Type Enumeration

| Secret Data Type | |
|---|---|
| **Name** | **Value** |
| Password | 00000001 |
| Seed | 00000002 |
| Extensions | 8XXXXXXX |

## 8 Opaque Data Type Enumeration

| Opaque Data Type | |
|---|---|
| **Name** | **Value** |
| Extensions | 8XXXXXXX |

## 9 Name Type Enumeration

| Name Type | |
|---|---|
| **Name** | **Value** |
| Uninterpreted Text String | 00000001 |
| URI | 00000002 |
| Extensions | 8XXXXXXX |

## 10 Object Type Enumeration

| Object Type | |
|---|---|
| **Name** | **Value** |
| Certificate | 00000001 |
| Symmetric Key | 00000002 |
| Public Key | 00000003 |

 

| | |
|---|---|
| Private Key | 00000004 |
| Split Key | 00000005 |
| Template | 00000006 |
| Policy Template | 00000007 |
| Secret Data | 00000008 |
| Opaque Object | 00000009 |
| Extensions | 8XXXXXXX |

## 11 Cryptographic Algorithm Enumeration

| **Cryptographic Algorithm** | |
|---|---|
| **Name** | **Value** |
| DES | 00000001 |
| 3DES | 00000002 |
| AES | 00000003 |
| RSA | 00000004 |
| DSA | 00000005 |
| ECDSA | 00000006 |
| HMAC-SHA1 | 00000007 |
| HMAC-SHA256 | 00000008 |
| HMAC-SHA512 | 00000009 |
| HMAC-MD5 | 0000000A |
| DH | 0000000B |
| ECDH | 0000000C |
| Extensions | 8XXXXXXX |

## 12 Block Cipher Mode Enumeration

| **Block Cipher Mode** | |
|---|---|
| **Name** | **Value** |
| CBC | 00000001 |
| ECB | 00000002 |
| PCBC | 00000003 |
| CFB | 00000004 |
| OFB | 00000005 |
| CTR | 00000006 |
| CMAC | 00000007 |
| CCM | 00000008 |

 

| | |
|---|---|
| GCM | 00000009 |
| CBC-MAC | 0000000A |
| AESKeyWrap | 0000000B |
| Extensions | 8XXXXXXX |

## 13 Padding Method Enumeration

| **Padding Method** | |
|---|---|
| **Name** | **Value** |
| None | 00000001 |
| OAEP | 00000002 |
| PKCS5 | 00000003 |
| SSL3 | 00000004 |
| Zeros | 00000005 |
| ANSI X9.23 | 00000006 |
| ISO 10126 | 00000007 |
| PKCS1 v1.5 | 00000008 |
| Extensions | 8XXXXXXX |

## 14 Hashing Algorithm Enumeration

| **Hashing Algorithm** | |
|---|---|
| **Name** | **Value** |
| MD2 | 00000001 |
| MD4 | 00000002 |
| MD5 | 00000003 |
| SHA-1 | 00000004 |
| SHA-256 | 00000005 |
| SHA-384 | 00000006 |
| SHA-512 | 00000007 |
| SHA-224 | 00000008 |
| Extensions | 8XXXXXXX |

## 15 Role Type Enumeration

| **Role Type** | |
|---|---|
| **Name** | **Value** |
| ZMK | 00000001 |

 

| | |
|---|---|
| ZPK | 00000002 |
| MAC | 00000003 |
| CVK | 00000004 |
| CSC | 00000005 |
| PVKIBM | 00000006 |
| PVKPVV | 00000007 |
| MKCVC | 00000008 |
| MKSMI | 00000009 |
| MKSMC | 0000000A |
| MKIDN | 0000000B |
| MKAC | 0000000C |
| MKCAP | 0000000D |
| BDK | 0000000E |
| Extensions | 8XXXXXXX |

## 16 State Enumeration

| **State** | |
|---|---|
| **Name** | **Value** |
| Pre-Active | 00000001 |
| Active | 00000002 |
| Deactivated | 00000003 |
| Compromised | 00000004 |
| Destroyed | 00000005 |
| Destroyed Compromised | 00000006 |
| Extensions | 8XXXXXXX |

## 17 Revocation Reason Code Enumeration

| **Revocation Reason Code** | |
|---|---|
| **Name** | **Value** |
| Key Compromise | 00000001 |
| CA Compromise | 00000002 |
| Affiliation Changed | 00000003 |
| Superseded | 00000004 |
| Cessation of Operation | 00000005 |
| Certificate Hold | 00000006 |
| Privilege Withdrawn | 00000007 |

 

| | |
|---|---|
| Revoked By creator | 00000008 |
| Revoked By Administrator | 00000009 |
| Extensions | 8XXXXXXX |

## 18 Link Type Enumeration

| Link Type | |
|---|---|
| **Name** | **Value** |
| Certificate Link | 00000101 |
| Public Key Link | 00000102 |
| Private Key Link | 00000103 |
| Derivation Base Object Link | 00000104 |
| Derived Key Link | 00000105 |
| Replacement Object Link | 00000106 |
| Replaced Object Link | 00000107 |
| Extensions | 8XXXXXXX |

Note: Link Types start at 101 to avoid any confusion with Object Types.

## 19 Derivation Method Enumeration

| Derivation Method | |
|---|---|
| **Name** | **Value** |
| PBKDF2 | 00000001 |
| HASH | 00000002 |
| HMAC | 00000003 |
| ENCRYPT | 00000004 |
| NIST800-108-C | 00000005 |
| NIST800-108-F | 00000006 |
| NIST800-108-DPI | 00000007 |
| Extensions | 8XXXXXXX |

## 20 Certificate Request Type Enumeration

| Certificate Request Type | |
|---|---|
| **Name** | **Value** |
| PCKS#10 | 00000001 |
| PEM | 00000002 |
| PGP | 00000003 |
| Extensions | 8XXXXXXX |

 

## 21 Validity Indicator Enumeration

| Validity Indicator | |
|---|---|
| **Name** | **Value** |
| Valid | 00000001 |
| Invalid | 00000002 |
| Unknown | 00000003 |
| Extensions | 8XXXXXXX |

## 22 Query Function Enumeration

| Query Function | |
|---|---|
| **Name** | **Value** |
| Query Operations | 00000001 |
| Query Objects | 00000002 |
| Query Server Information | 00000003 |
| Extensions | 8XXXXXXX |

## 23 Cancellation Result Enumeration

| Cancellation Result | |
|---|---|
| **Name** | **Value** |
| Canceled | 00000001 |
| Unable to Cancel | 00000002 |
| Completed | 00000003 |
| Failed | 00000004 |
| Unavailable | 00000005 |
| Extensions | 8XXXXXXX |

## 24 Put Function Enumeration

| Put Function | |
|---|---|
| **Name** | **Value** |
| New | 00000001 |
| Replace | 00000002 |
| Extensions | 8XXXXXXX |

 

## 25 Operations Enumeration

| Operation | |
|---|---|
| **Name** | **Value** |
| Create | 00000001 |
| Create Key Pair | 00000002 |
| Register | 00000003 |
| Re-key | 00000004 |
| Derive Key | 00000005 |
| Certify | 00000006 |
| Re-certify | 00000007 |
| Locate | 00000008 |
| Check | 00000009 |
| Get | 0000000A |
| Get Attributes | 0000000B |
| Get Attribute List | 0000000C |
| Add Attribute | 0000000D |
| Modify Attribute | 0000000E |
| Delete Attribute | 0000000F |
| Obtain Lease | 00000010 |
| Get Usage Allocation | 00000011 |
| Activate | 00000012 |
| Revoke | 00000013 |
| Destroy | 00000014 |
| Archive | 00000015 |
| Recover | 00000016 |
| Validate | 00000017 |
| Query | 00000018 |
| Cancel | 00000019 |
| Poll | 0000001A |
| Notify | 0000001B |
| Put | 0000001C |
| Extensions | 8XXXXXXX |

## 26 Result Status Enumeration

| Result Status | |
|---|---|
| **Name** | **Value** |

 

| | |
|---|---|
| Success | 00000000 |
| Operation Failed | 00000001 |
| Operation Pending | 00000002 |
| Operation Undone | 00000003 |
| Extensions | 8XXXXXXX |

### 27 Result Reason Enumeration

| Result Reason | |
|---|---|
| **Name** | **Value** |
| Item Not Found | 00000001 |
| Response Too Large | 00000002 |
| Authentication Not Successful | 00000003 |
| Invalid Message | 00000004 |
| Operation Not Supported | 00000005 |
| Missing Data | 00000006 |
| Invalid Field | 00000007 |
| Feature Not Supported | 00000008 |
| Operation Canceled By Requester | 00000009 |
| Cryptographic Failure | 0000000A |
| Illegal Operation | 0000000B |
| Permission Denied | 0000000C |
| Object archived | 0000000D |
| General Failure | 00000100 |
| Extensions | 8XXXXXXX |

### 28 Batch Error Continuation Enumeration

| Batch Error Continuation | |
|---|---|
| **Name** | **Value** |
| Continue | 00000001 |
| Stop | 00000002 |
| Undo | 00000003 |
| Extensions | 8XXXXXXX |

## *3 Bit Masks*

### 1 Cryptographic Usage Mask Values

 

| Cryptographic Usage Mask | |
|---|---|
| **Name** | **Value** |
| Sign | 00000001 |
| Verify | 00000002 |
| Encrypt | 00000004 |
| Decrypt | 00000008 |
| Wrap | 00000010 |
| Unwrap | 00000020 |
| Export | 00000040 |
| MAC | 00000080 |
| Derive Key | 00000100 |
| Content Commitment (Non Repudiation) | 00000200 |
| Key Agreement | 00000400 |
| Certificate Sign | 00000800 |
| CRL Sign | 00001000 |
| MAC Verify | 00002000 |
| Extensions | XXXX0000 |

This list takes into consideration values which may appear in the Key Usage extension in an X.509 certificate. However, the list does not consider the more fined grained usages which may appear in the Extended Key Usage extension.

### 2 Storage Status Mask

| Storage Status Values | |
|---|---|
| **Name** | **Value** |
| On-line storage | 00000001 |
| Archival storage | 00000002 |
| Extensions | XXXXXXX0 |

## 2 XML Encoding

An XML Encoding has not yet been defined.

# 9 Transport

Transport protocols are not part of the message definitions, and are external to this protocol. The Usage Guide, however, describes two profiles for implementation of this protocol over secure transport protocols, namely:

- SSL/TLS over TCP. This profile describes the implementation of this protocol using SSL/TLS encryption, with client and server authentication features enabled, over a normal TCP stream.

 

- HTTPS over TCP. This profile describes the implementation of this protocol using HTTPS, with client and server authentication features enabled, over a normal TCP stream.

To ensure a base level of interoperability, all server implementations should, at least, support the SSL/TLS and HTTPS transport protocols as described in the Usage Guide.

# 10 Error Handling

This section details the specific Result Reasons that should be returned for errors detected. Note that this is not an exhaustive list of possible errors for each operation (allowing other Result Reasons to be returned if an implementation needs to do so).

## 1 General

These errors may occur when any protocol message is received by the server.

| Error Definition | Action | Result Reason |
|---|---|---|
| Protocol major version mismatch | Response message containing a header and a Batch Item without Operation but with the Result Status field set to Operation Failed | Invalid Message |
| Error parsing batch item or payload within batch item (required fields missing, etc.) | Batch item fails, Result Status is Operation Failed | Invalid Message |
| The same field is contained in a header/batch item/payload more than once | Result Status is Operation Failed | Invalid Message |
| Same major version, different minor versions (client is newer), unknown fields/fields the server does not understand | Ignore unknown fields, process rest normally | N/A |
| Same major & minor version, unknown field | Result Status is Operation Failed | Invalid Field |
| Client is not allowed to perform the specified operation | Result Status is Operation Failed | Permission Denied |
| Operation cannot be completed synchronously and client does not support asynchronous requests | Result Status is Operation Failed | Operation Not Supported |
| Maximum Response Size has been exceeded | Result Status is Operation Failed | Response Too Large |

## 2 Create

| Error Definition | Result Status | Result Reason |
|---|---|---|
| Object Type is not recognized | Operation Failed | Invalid Field |
| Templates that do not exist are given in request | Operation Failed | Item Not Found |
| Incorrect attribute value(s) specified (e.g. initial date 5 years ago) | Operation Failed | Invalid Field |
| Error creating cryptographic object (key material generation issue) | Operation Failed | Cryptographic Failure |
| Trying to create a new object with the same Name attribute value as an existing object | Operation Failed | Invalid Field |

 

## 3 Create Key Pair

| Error Definition | Result Status | Result Reason |
|---|---|---|
| Templates that do not exist are given in request | Operation Failed | Item Not Found |
| Incorrect attribute value(s) specified | Operation Failed | Invalid Field |
| Error creating cryptographic object (key material generation issue) | Operation Failed | Cryptographic Failure |
| Trying to create a new object with the same Name attribute value as an existing object | Operation Failed | Invalid Field |
| Required field(s) missing | Operation Failed | Invalid Message |

## 4 Register

| Error Definition | Result Status | Result Reason |
|---|---|---|
| Object Type is not recognized | Operation Failed | Invalid Field |
| Object Type does not match type of cryptographic object provided | Operation Failed | Invalid Field |
| Templates that do not exist are given in request | Operation Failed | Item Not Found |
| Incorrect attribute value(s) specified (e.g. initial date 5 years ago) | Operation Failed | Invalid Field |
| Trying to register a new object with the same Name attribute value as an existing object | Operation Failed | Invalid Field |

## 5 Re-key

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Object specified cannot be re-keyed (not a symmetric key or the permissions do not allow it) | Operation Failed | Permission Denied |
| Offset field cannot be specified at the same time as any of the Activation Date, Process Start Date, Protect Stop Date, or Deactivation Date attributes | Operation Failed | Invalid Message |
| Cryptographic error during re-key | Operation Failed | Cryptographic Failure |
| Object is archived | Operation Failed | Object archived |

## 6 Derive Key

| Error Definition | Result Status | Result Reason |
|---|---|---|
| One or more of the objects specified do not | Operation Failed | Item Not Found |

 

| | | |
|---|---|---|
| exist | | |
| One or more of the objects specified are not of the correct type | Operation Failed | Invalid Field |
| Templates that do not exist are given in request | Operation Failed | Item Not Found |
| Invalid Derivation Method | Operation Failed | Invalid Field |
| Invalid Derivation Parameters | Operation Failed | Invalid Field |
| Ambiguous derivation data provided both with Derivation Data and Secret Data object. | Operation Failed | Invalid Message |
| Incorrect attribute value(s) specified (e.g. initial date 5 years ago) | Operation Failed | Invalid Field |
| One or more of the specified objects cannot be used to derive a new key | Operation Failed | Invalid Field |
| Trying to derive a new key with the same Name attribute value as an existing object | Operation Failed | Invalid Field |
| One or more of the objects is archived | Operation Failed | Object archived |

# 7 Certify

| **Error Definition** | **Result Status** | **Result Reason** |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Object specified cannot be certified (not a public key or the permissions do not allow it) | Operation Failed | Permission Denied |
| The Certificate Request does not contain a signed certificate request of the specified Certificate Request Type | Operation Failed | Invalid Field |
| Server does not support operation | Operation Failed | Operation Not Supported |
| Object is archived | Operation Failed | Object archived |

# 8 Re-certify

| **Error Definition** | **Result Status** | **Result Reason** |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Object specified cannot be certified (not a certificate or the permissions do not allow it) | Operation Failed | Permission Denied |
| The Certificate Request does not contain a signed certificate request of the specified Certificate Request Type | Operation Failed | Invalid Field |
| Server does not support operation | Operation Failed | Operation Not Supported |
| Offset field cannot be specified at the same time as any of the Activation Date or Deactivation Date attributes | Operation Failed | Invalid Message |
| Object is archived | Operation Failed | Object archived |

 

## 9 Locate

| Error Definition | Result Status | Result Reason |
|---|---|---|
| Non-existing attributes, attributes that the server does not understand or templates that do not exist are given in request | Operation Failed | Invalid Field |

## 10 Check

| Error Definition | Result Status | Result Reason |
|---|---|---|
| Object does not exist | Operation Failed | Item Not Found |
| Object is archived | Operation Failed | Object archived |

## 11 Get

| Error Definition | Result Status | Result Reason |
|---|---|---|
| Object does not exist | Operation Failed | Item Not Found |
| Wrapping key does not exist | Operation Failed | Item Not Found |
| Object with Wrapping Key ID exists but it is not a key | Operation Failed | Illegal Operation |
| Object with Wrapping Key ID exists but it cannot be used for wrapping | Operation Failed | Permission Denied |
| Object with MAC/Signature Key ID exists but it is not a key | Operation Failed | Illegal Operation |
| Object with MAC/Signature Key ID exists but it cannot be used for MAc'ing/signing | Operation Failed | Permission Denied |
| No cryptographic material associated with object | Operation Failed | Illegal Operation |
| Cryptographic Parameters associated with object do not exist or do not match with those provided in the Encryption Key Information and/or Signature Key Information | Operation Failed | Item Not Found |
| Object is archived | Operation Failed | Object archived |

## 12 Get Attributes

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Object is archived | Operation Failed | Object archived |

## 13 Get Attribute List

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |

 

| | | |
|---|---|---|
| Object is archived | Operation Failed | Object archived |

## 14 Add Attribute

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Attempt to add read-only attribute | Operation Failed | Permission Denied |
| The specified attribute already exists | Operation Failed | Illegal Operation |
| New attribute contains index | Operation Failed | Invalid Field |
| Trying to add a Name attribute with the same value that another object already has | Operation Failed | Illegal Operation |
| Object is archived | Operation Failed | Object archived |

## 15 Modify Attribute

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| A specified attribute does not exist (must first be added) | Operation Failed | Invalid Field |
| An Attribute Index is specified but no matching instance exists. | Operation Failed | Item Not Found |
| The specified attribute is read-only | Operation Failed | Permission Denied |
| Trying to set the Name attribute value to something that another object already has | Operation Failed | Illegal Operation |
| Object is archived | Operation Failed | Object archived |

## 16 Delete Attribute

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Attempt to delete read-only/required attribute | Operation Failed | Permission Denied |
| Attribute index is specified but attribute does not have multiple instances and therefore no index | Operation Failed | Item Not Found |
| No attribute with specified name exists | Operation Failed | Item Not Found |
| Object is archived | Operation Failed | Object archived |

## 17 Obtain Lease

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| The server determines that a new lease | Operation Failed | Permission Denied |

 

| | | |
|---|---|---|
| should not be issued for the specified cryptographic object | | |
| Object is archived | Operation Failed | Object archived |

## 18 Get Usage Allocation

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Object has no Usage Limits attribute or object cannot be used for protection purposes | Operation Failed | Illegal Operation |
| Both Usage Limits Byte Count and Usage Limits Object Count fields specified | Operation Failed | Invalid Message |
| Neither Byte Count or Object Count is specified | Operation Failed | Invalid Message |
| A usage type (Byte Count or Object Count) is specified in the request, but the usage allocation for the object can only be given for the other type | Operation Failed | Operation Not Supported |
| Object is archived | Operation Failed | Object archived |

## 19 Activate

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Unique Identifier specifies template, policy template or other object that cannot be revoked | Operation Failed | Illegal Operation |
| Object is not in Pre-Active state | Operation Failed | Permission Denied |
| Object is archived | Operation Failed | Object archived |

## 20 Revoke

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Revocation Reason is not recognized | Operation Failed | Invalid Field |
| Unique Identifier specifies template, policy template or other object that cannot be revoked | Operation Failed | Illegal Operation |
| Object is archived | Operation Failed | Object archived |

 

## 21 Destroy

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Object exists but has already been destroyed | Operation Failed | Permission Denied |
| Object is not in Deactivated state | Operation Failed | Permission Denied |
| Object is archived | Operation Failed | Object archived |

## 22 Archive

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |
| Object is already archived | Operation Failed | Object archived |

## 23 Recover

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No object with the specified Unique Identifier exists | Operation Failed | Item Not Found |

## 24 Validate

| Error Definition | Result Status | Result Reason |
|---|---|---|
| The combination of Certificate Objects and Unique Identifiers do not specify a certificate list | Operation Failed | Invalid Message |
| One or more of the objects is archived | Operation Failed | Object archived |

## 25 Query

N/A

## 26 Cancel

N/A

## 27 Poll

| Error Definition | Result Status | Result Reason |
|---|---|---|
| No outstanding operation with the specified Asynchronous Correlation Value exists | Operation Failed | Item Not Found |

 

## 28 Batch Items

These errors may occur when a protocol message with one or more batch items is processed by the server. If a message with one or more than a single batch item was parsed correctly, the response message should include response(s) to the batch item(s) in the request according to the table below.

| Error Definition | Result Status | Result Reason |
|---|---|---|
| Processing of batch item fails with Batch Error Continuation Option set to Stop | Batch item fails. Responses to batch items that have already been processed are returned normally. Responses to batch items that have not been processed are not returned. | See tables above, referring to the operation being performed in the batch item that failed |
| Processing of batch item fails with Batch Error Continuation Option set to Continue | Batch item fails. Responses to other batch items are returned normally. | See tables above, referring to the operation being performed in the batch item that failed |
| Processing of batch item fails with Batch Error Continuation Option set to Undo | Batch item fails. Batch items that had been processed have been undone and their responses are returned with Undone result status. | See tables above, referring to the operation being performed in the batch item that failed |

# 11 Attribute Cross-reference

The following table of Attribute names indicates the Managed Object(s) for which each attribute applies:

| Attribute Name | Managed Object | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Certificate | Symmetric Key | Public Key | Private Key | Split Key | Template | Policy Template | Secret Data | Opaque Object |
| Unique Identifier | x | x | x | x | x | x | x | x | x |
| Name | x | x | x | x | x | x | x | x | x |
| Object Type | x | x | x | x | x | x | x | x | x |
| Cryptographic Algorithm | x | x | x | x | x | | | | |
| Cryptographic Length | x | x | x | x | x | | | | |
| Cryptographic Parameters | x | x | x | x | x | | | | |
| Certificate Type | x | | | | | | | | |
| Certificate Issuer | x | | | | | | | | |
| Certificate Subject | x | | | | | | | | |
| Digest | x | x | x | x | x | | | | |
| Operation Policy Name | x | x | x | x | x | x | x | x | x |
| Cryptographic Usage Mask | x | x | x | x | x | | | | |
| Lease Time | x | x | x | x | x | | | x | x |

 

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Usage Limits | | x | x | x | x | | | x | x |
| State | x | x | x | x | x | | | | |
| Initial Date | x | x | x | x | x | x | x | x | x |
| Activation Date | x | x | x | x | x | x | x | x | x |
| Process Start Date | | x | x | x | | | | | |
| Protect Stop Date | | x | x | x | | | | | |
| Deactivation Date | x | x | x | x | x | | | | |
| Destroy Date | x | x | x | x | x | x | x | x | x |
| Compromise Occurrence Date | | x | x | x | x | | | | |
| Compromise Date | | x | x | x | x | | | | |
| Revocation Reason | x | x | x | x | x | | | | |
| Archive Date | x | x | x | x | x | x | x | x | x |
| Object Type | x | x | x | x | x | x | x | x | x |
| Link | x | x | x | x | x | x | x | x | x |
| Application Specific Identification | x | x | x | x | x | | | | |
| Contact Information | x | x | x | x | x | x | x | x | x |
| Last Changed Date | x | x | x | x | x | x | x | x | x |
| Custom Attribute | x | x | x | x | x | x | x | x | x |

 

# 12 Tag Cross-reference

| Object | Defined | Type | Notes |
|---|---|---|---|
| Activation Date | 3.17 | Date-Time | |
| Application Identifier | 3.28 | Text String | |
| Application Name Space | 3.28 | Text String | |
| Application Specific Identification | 3.28 | Structure | |
| Archive Date | 3.25 | Date-Time | |
| Asynchronous Correlation Value | 6.8 | Octet String | |
| Asynchronous Indicator | 6.7 | Boolean | |
| Attribute | 2.1.1 | Structure | |
| Attribute Index | 2.1.1 | Integer | |
| Attribute Name | 2.1.1 | Text String | |
| Attribute Value | 2.1.1 | * | type varies |
| Authentication | 6.6 | Structure | |
| Batch Count | 6.14 | Integer | |
| Batch Error Continuation Option | 6.13 , 9.1.3.2.28 | Enumeration | |
| Batch Item | 6.15 | Structure | |
| Batch Order Option | 6.12 | Boolean | |
| Block Cipher Mode | 3.6 , 9.1.3.2.12 | Enumeration | |
| Cancellation Result | 4.25 , 9.1.3.2.23 | Enumeration | |
| Certificate | 2.2.1 | Structure | |
| Certificate Issuer | 3.8 | Structure | |
| Certificate Request | 4.6 , 4.7 | Octet String | |
| Certificate Request Type | 4.6 , 4.7 , 9.1.3.2.20 | Enumeration | |
| Certificate Subject | 3.9 | Structure | |
| Certificate Subject Alternative Name | 3.9 | Text String | |
| Certificate Subject Distinguished Name | 3.9 | Text String | |
| Certificate Type | 2.2.1 , 3.7 , 9.1.3.2.5 | Enumeration | |
| Certificate Value | 2.2.1 | Octet String | |
| Common Template-Attribute | 2.1.8 | Structure | |
| Compromise Occurrence Date | **Error! Reference source not found.** | Date-Time | |
| Compromise Date | 3.23 | Date-Time | |
| Contact Information | 3.29 | Text String | |
| Credential | 2.1.2 | Structure | |
| Credential Type | 2.1.2 , 9.1.3.2.1 | Enumeration | |
| Credential Value | 2.1.2 | Octet String | |
| Criticality Indicator | 6.16 | Boolean | |
| CRT Coefficient | 2.1.7 | Big Integer | |
| Cryptographic Algorithm | 3.4 , 9.1.3.2.11 | Enumeration | |
| Cryptographic Length | 3.5 | Integer | |
| Cryptographic Parameters | 3.6 | Structure | |

 

| | | | |
|---|---|---|---|
| Cryptographic Usage Mask | 3.12 , 9.1.3.3.1 | Integer | Bit mask |
| Custom Attribute | 3.31 | * | type varies |
| D | 2.1.7 | Big Integer | |
| Deactivation Date | 3.20 | Date-Time | |
| Derivation Data | 4.5 | Octet String | |
| Derivation Method | 4.5 , 9.1.3.2.19 | Enumeration | |
| Derivation Parameters | 4.5 | Structure | |
| Destroy Date | 3.21 | Date-Time | |
| Digest | 3.10 | Structure | |
| Digest Value | 3.10 | Octet String | |
| Encryption Key Information | 2.1.5 | Structure | |
| Extensions | 9.1.3 | | |
| G | 2.1.7 | Big Integer | |
| Hashing Algorithm | 3.6 , 3.10 , 9.1.3.2.14 | Enumeration | |
| Initial Date | 3.16 | Date-Time | |
| Initialization Vector | 4.5 | Octet String | |
| Issuer | 3.8 | Text String | |
| Iteration Count | 4.5 | Integer | |
| IV/Counter/Nonce | 2.1.5 | Octet String | |
| J | 2.1.7 | Big Integer | |
| Key | 2.1.7 | Octet String | |
| Key Block | 2.1.3 | Structure | |
| Key Material | 2.1.4 , 2.1.7 | Octet String / Structure | |
| Key Part Identifier | 2.2.5 | Integer | |
| Key Value | 2.1.4 | Octet String / Structure | |
| Key Value Type | 2.1.4 , 9.1.3.2.2 | Enumeration | |
| Key Wrapping Data | 2.1.5 | Structure | |
| Key Wrapping Specification | 2.1.6 | Structure | |
| Last Changed Date | 3.30 | Date-Time | |
| Lease Time | 3.13 | Interval | |
| Link | 3.27 | Structure | |
| Link Type | 3.27 , 9.1.3.2.18 | Enumeration | |
| Linked Object Identifier | 3.27 | Text String | |
| MAC/Signature | 2.1.5 | Octet String | |
| MAC/Signature Key Information | 2.1.5 | Text String | |
| Maximum Items | 4.8 | Integer | |
| Maximum Response Size | 6.3 | Integer | |
| Message Extension | 6.16 | Structure | |
| Modulus | 2.1.7 | Big Integer | |
| Name | 3.2 | Structure | |
| Name Type | 3.2 , 9.1.3.2.9 | Enumeration | |
| Name Value | 3.2 | Text String | |
| Object Group | 3.26 | Text String | |

 

| | | | |
|---|---|---|---|
| Object Type | 3.3 , 9.1.3.2.10 | Enumeration | |
| Offset | 4.4 , 4.7 | Interval | |
| Opaque Data Type | 2.2.9 , 9.1.3.2.8 | Enumeration | |
| Opaque Data Value | 2.2.9 | Octet String | |
| Opaque Object | 2.2.9 | Structure | |
| Operation | 6.2 , 9.1.3.2.25 | Enumeration | |
| Operation Policy Name | 3.11 | Text String | |
| P | 2.1.7 | Big Integer | |
| Padding Method | 3.6 , 9.1.3.2.13 | Enumeration | |
| Policy Template | 2.2.7 | Structure | |
| Prime Exponent P | 2.1.7 | Big Integer | |
| Prime Exponent Q | 2.1.7 | Big Integer | |
| Prime Field Size | 2.2.5 | Big Integer | |
| Private Exponent | 2.1.7 | Big Integer | |
| Private Key | 2.2.4 | Structure | |
| Private Key Template-Attribute | 2.1.8 | Structure | |
| Private Key Unique Identifier | 4.2 | Text String | |
| Process Start Date | 3.18 | Date-Time | |
| Protect Stop Date | 3.19 | Date-Time | |
| Protocol Version | 6.1 | Structure | |
| Protocol Version Major | 6.1 | Integer | |
| Protocol Version Minor | 6.1 | Integer | |
| Public Exponent | 2.1.7 | Big Integer | |
| Public Key | 2.2.3 | Structure | |
| Public Key Template-Attribute | 2.1.8 | Structure | |
| Public Key Unique Identifier | 4.2 | Text String | |
| Put Function | 5.2 , 9.1.3.2.24 | Enumeration | |
| Q | 2.1.7 | Big Integer | |
| Q String | 2.1.7 | Octet String | |
| Query Function | 4.24 , 9.1.3.2.22 | Enumeration | |
| Recommended Curve | 2.1.7 , 9.1.3.2.4 | Enumeration | |
| Replaced Unique Identifier | 5.2 | Text String | |
| Request Header | 7.2 , 7.3 | Structure | |
| Request Message | 7.1 | Structure | |
| Request Payload | 4 , 5 , 7.2 , 7.3 | Structure | |
| Response Header | 7.2 , 7.3 | Structure | |
| Response Message | 7.1 | Structure | |
| Response Payload | 4 , 7.2 , 7.3 | Structure | |
| Result Message | 6.11 | Text String | |
| Result Reason | 6.10 , 9.1.3.2.27 | Enumeration | |
| Result Status | 6.9 , 9.1.3.2.26 | Enumeration | |
| Revocation Message | 3.24 | Text String | |
| Revocation Reason | 3.24 | Structure | |
| Revocation Reason Code | 3.24 , 9.1.3.2.17 | Enumeration | |

 

| | | | |
|---|---|---|---|
| Role Type | 3.6 , 9.1.3.2.15 | Enumeration | |
| Salt | 4.5 | Octet String | |
| Secret Data | 2.2.8 | Structure | |
| Secret Data Type | 2.2.8 , 9.1.3.2.7 | Enumeration | |
| Serial Number | 3.8 | Text String | |
| Server Information | 4.24 | Structure | contents vendor-specific |
| Split Key | 2.2.5 | Structure | |
| Split Key Method | 2.2.5 , 9.1.3.2.6 | Enumeration | |
| Split Key Parts | 2.2.5 | Integer | |
| Split Key Threshold | 2.2.5 | Integer | |
| State | 3.15 , 9.1.3.2.16 | Enumeration | |
| Storage Status Mask | 4.8 , 9.1.3.3.2 | Integer | Bit mask |
| Symmetric Key | 2.2.2 | Structure | |
| Template | 2.2.6 | Structure | |
| Template Name | 4.3 | Text String | |
| Template-Attribute | 2.1.8 | Structure | |
| Time Stamp | 6.5 | Date-Time | |
| Transparent* | 2.1.7 | Structure | |
| Unique Identifier | 3.1 | Text String | |
| Unique Message ID | 6.4 | Octet String | |
| Usage Limits | 3.14 | Structure | |
| Usage Limits Byte Count | 3.14 | Big Integer | |
| Usage Limits Object Count | 3.14 | Big Integer | |
| Usage Limits Total Bytes | 3.14 | Big Integer | |
| Usage Limits Total Objects | 3.14 | Big Integer | |
| Validity Date | 4.23 | Date-Time | |
| Validity Indicator | 4.23 , 9.1.3.2.21 | Enumeration | |
| Vendor Extension | 6.16 | Structure | contents vendor-specific |
| Vendor Identification | 4.24 , 6.16 | Text String | |
| Wrapping Method | 2.1.5 , 9.1.3.2.3 | Enumeration | |
| X | 2.1.7 | Big Integer | |
| Y | 2.1.7 | Big Integer | |

 

# 13 Acronyms

The following abbreviations and acronyms are used in this document:

| | |
|---|---|
| 3DES | - Three key Data Encryption Standard |
| AES | - Advanced Encryption Standard specified in FIPS 197 |
| ASN.1 | - Abstract Syntax Notation One |
| CA | - Certification Authority |
| CBC | - Cipher Block Chaining |
| CPU | - Central Processing Unit |
| CRL | - Certificate Revocation List |
| CRT | - Chinese Remainder Theorem |
| DER | - Distinguished Encoding Rules |
| DES | - Data Encryption Standard |
| DH | - Diffie-Hellman |
| DSA | - Digital Signature Algorithm specified in FIPS 186-3 |
| DSKPP | - Dynamic Symmetric Key Provisioning Protocol |
| ECB | - Electronic Code Book |
| ECDH | - Elliptic Curve Diffie-Hellman |
| ECDSA | - Elliptic Curve Digital Signature Algorithm specified in ANSX9.62 |
| HMAC | - Keyed-Hash Message Authentication Code specified in FIPS 198 |
| HTTP | - Hyper Text Transfer Protocol |
| HTTP(S) | - Hyper Text Transfer Protocol (Secure socket) |
| IEEE | - Institute of Electrical and Electronics Engineers |
| IETF | - Internet Engineering Task Force |
| IPsec | - Internet Protocol Security |
| IV | - Initialization Vector |
| KMIP | - Key Management Interoperability Protocol |
| MAC | - Message Authentication Code |
| MD5 | - Message Digest 5 Algorithm |
| PBKDF2 | - Password-Based Key Derivation Function 2 |
| PGP | - Pretty Good Privacy |
| PKCS | - Public Key Cryptography Standards |
| POSIX | - Portable Operating System Interface |
| RFC | - Request for Comments documents of IETF |
| RSA | - Rivest, Shamir, Adelman (an algorithm) |
| SHA-1 | - Secure Hash Algorithm Revision One |
| SSL/TLS | - Secure Sockets Layer/Transport Layer Security |
| S/MIME | - Secure/Multipurpose Internet Mail Extensions |
| TCP | - Transport Control Protocol |
| TTLV | - Tag, Type, Length, Value |
| URI | - Unique Resource Identifier |
| UTF | - Universal Transformation Format |
| XML | - Extensible Markup Language |

 

# 14 Acknowledgments

The following people (in alphabetical order) contributed to this document:

- David Babcock, HP
- Steven Bade, IBM
- Paolo Bezoari, NetApp
- Mathias Björkqvist, IBM
- Bruce Brinson, EMC
- Christian Cachin, IBM
- Tony Crossman, nCipher
- Stan Feather, HP
- Indra Fitzgerald, HP
- Judy Furlong, EMC
- Jon Geater, nCipher
- Bob Griffin, EMC
- Robert Haas, IBM (editor)
- Timothy Hahn, IBM
- Jack Harwood, EMC
- Walt Hubis, LSI
- Glen Jaquette, IBM
- Jeff Kravitz, IBM (editor)
- Michael McIntosh, IBM
- Brian Metzger, HP
- Anthony Nadalin, IBM
- Elaine Palmer, IBM
- Joe Pato, HP
- René Pawlitzek, IBM
- Subhash Sankuratripati, NetApp
- Mark Schiller, HP
- Martin Skagen, Brocade
- Marcus Streets, nCipher
- John Tattan, EMC
- Karla Thomas, Brocade
- Marko Vukolić , IBM
- Steve Wierenga, HP

 