Key Management Interoperability Protocol Specification Version 1.0

Committee Draft 07

Deleted: 06 / Public Review 01

04 February 2010

Deleted: 05 November 2009

Specification URI

This Version:

http://docs.oasis-open.org/kmip/spec/v1.0/cd07/kmip-spec-1.0-cd-07.html
http://docs.oasis-open.org/kmip/spec/v1.0/cd07/kmip-spec-1.0-cd-07.doc (Authoritative)
http://docs.oasis-open.org/kmip/spec/v1.0/cd07/kmip-spec-1.0-cd-07.pdf

Previous Version:

N/A

Latest Version:

http://docs.oasis-open.org/kmip/spec/v1.0/kmip-spec-1.0.html
http://docs.oasis-open.org/kmip/spec/v1.0/kmip-spec-1.0.doc
http://docs.oasis-open.org/kmip/spec/v1.0/kmip-spec-1.0.pdf

Technical Committee:

OASIS Key Management Interoperability Protocol (KMIP) TC

Chair(s):

Robert Griffin, EMC Corporation <robert.griffin@rsa.com>
Subhash Sankuratripati, NetApp <Subhash.Sankuratripati@netapp.com>

Editor(s):

Robert Haas, IBM <rha@zurich.ibm.com>
Indra Fitzgerald, HP <indra.fitzgerald@hp.com>

Related work:

This specification replaces or supersedes:

- None

This specification is related to:

- Key Management Interoperability Protocol Profiles Version 1.0
- Key Management Interoperability Protocol Use Cases Version 1.0
- Key Management Interoperability Protocol Usage Guide Version 1.0

Declared XML Namespace(s):

None

Abstract:

This document is intended for developers and architects who wish to design systems and applications that interoperate using the Key Management Interoperability Protocol specification.

Status:

This document was last revised or approved by the Key Management Interoperability Protocol TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the

"Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/kmip/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/kmip/ipr.php).

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/kmip/.

Notices

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 are 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 are obtained from the system via operations. Certain attributes are added, modified, or deleted 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 needed for a certification authority. However, it does include functions that are needed to allow a key server to provide a proxy for certificate management functions.

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:

- The expected behavior of the server and client as a result of operations,
- Message contents and formats,
- Message encoding (including enumerations), and
- Error handling.

Deleted:

This specification is complemented by three other documents. The Usage Guide **[KMIP-UG]** provides illustrative information on using the protocol. The KMIP Profiles Specification **[KMIP-Prof]** provides a selected set of conformance profiles and authentication suites. The Test Specification **[KMIP-UC]** provides samples of protocol messages corresponding to a set of defined test cases.

This specification defines the KMIP protocol version major 1 and minor 0 (see 6.1).

Deleted: 6.1

1.1 Terminology

The key words "SHALL", "SHALL NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. The words ‘must’, ‘can’, and ‘will’ are forbidden.

For definitions not found in this document, see **[SP800-57-1]**.

Archive	To place information not accessed frequently into long-term storage
Asymmetric key pair (key pair)	A public key and its corresponding private key; a key pair is used with a public key algorithm
Authentication	A process that establishes the origin of information, or determines an entity’s identity.
Authentication code	A cryptographic checksum based on an approved security function (also known as a Message Authentication Code).
Authorization	Access privileges that are granted to an entity; conveying an “official” sanction to perform a security function or activity.

Deleted: A

Certification authority	The entity in a Public Key Infrastructure (PKI) that is responsible for issuing certificates, and exacting compliance to a PKI policy.
Ciphertext	Data in its encrypted form.
Compromise	The unauthorized disclosure, modification, substitution or use of sensitive data (e.g., keying material and other security-related information).
Confidentiality	The property that sensitive information is not disclosed to unauthorized entities.
Cryptographic algorithm	A well-defined computational procedure that takes variable inputs, including a cryptographic key and produces an output.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Examples include: 1. The transformation of plaintext data into ciphertext data, 2. The transformation of ciphertext data into plaintext data, 3. The computation of a digital signature from data, 4. The verification of a digital signature, 5. The computation of an authentication code from data, 6. The verification of an authentication code from data and a received authentication code,
Decryption	The process of changing ciphertext into plaintext using a cryptographic algorithm and key.
Digest (or hash)	The result of applying a hash function to information.
Digital signature (signature)	The result of a cryptographic transformation of data that, when properly implemented with supporting infrastructure and policy, provides the services of: 1. origin authentication 2. data integrity, and 3. signer non-repudiation.
Encryption	The process of changing plaintext into ciphertext using a cryptographic algorithm and key.
Hashing algorithm	An algorithm that maps a bit string of arbitrary length to a fixed length bit string. Approved hashing algorithms satisfy the following properties: 1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and 2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output.
Integrity	The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner.
Key derivation (derivation)	A function in the lifecycle of keying material; the process by which one or more keys are derived from a shared secret and other information.

Deleted:

Deleted: function

Deleted: function

Deleted: functions

Comment: EBB: This depends on how a hash function is used. It is required for digital signatures, but not for HMAC or KDFs, for example.
I think NIST may need to work on this.

Comment: MBJ: proposed resolution: no change

Comment: EBB: This is a definition when used during key agreement. Another use is to derive additional keys from a preshared key.
NIST needs to work on this.

Comment: MBJ: proposed resolution: no change

Key management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., IVs and passwords) during the entire life cycle of the keys, including their generation, storage, establishment, entry and output, and destruction.
Key wrapping (wrapping)	A method of encrypting and/or MACing/signing keys using cryptographic keys.
Message authentication code (MAC)	A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data.
Private key	A cryptographic key, used with a public key cryptographic algorithm, that is uniquely associated with an entity and is not made public. In an asymmetric (public) cryptosystem, the private key is associated with a public key. Depending on the algorithm, the private key may be used to: 1. Compute the corresponding public key, 2. Compute a digital signature that may be verified by the corresponding public key, 3. Decrypt data that was encrypted by the corresponding public key, or 4. Compute a piece of common shared data, together with other information.
Profile	A specification of objects, attributes, operations, message elements and authentication methods to be used in specific contexts of key management server and client interactions (see **[KMIP-Prof]**).
Public key	A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that may be made public. In an asymmetric (public) cryptosystem, the public key is associated with a private key. The public key may be known by anyone and, depending on the algorithm, may be used to: 1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt data that can be decrypted by the corresponding private key, or 3. Compute a piece of shared data.
Public key certificate (certificate)	A set of data that uniquely identifies an entity, contains the entity's public key and possibly other information, and is digitally signed by a trusted party, thereby binding the public key to the entity.
Public key cryptographic algorithm	A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible.
Public Key Infrastructure	A framework that is established to issue, maintain and revoke public key certificates.
Recover	To retrieve information that was archived to long-term storage.
Split knowledge	A process by which a cryptographic key is split into *n* multiple key components, individually providing no knowledge of the original key, which can be subsequently combined to recreate the original cryptographic key. If knowledge of *k* (where *k* is less than or equal to *n*) components is required to construct the original key, then knowledge of any *k*-1 key components provides no information about the original key

Deleted: (along with associated integrity information)

Deleted: that provides both confidentiality and integrity protection

Deleted: a

Deleted: symmetric

	other than, possibility, its length.
Symmetric key	A single cryptographic key that is used with a secret (symmetric) key algorithm.
Symmetric key algorithm	A cryptographic algorithm that uses the same secret (symmetric) key for an operation and its complement (e.g., encryption and decryption).
X.509 certificate	The ISO/ITU-T X.509 standard defined two types of certificates – the X.509 public key certificate, and the X.509 attribute certificate. Most commonly (including this document), an X.509 certificate refers to the X.509 public key certificate.
X.509 public key certificate	The public key for a user (or device) and a name for the user (or device), together with some other information, rendered un-forgeable by the digital signature of the certification authority that issued the certificate, encoded in the format defined in the ISO/ITU-T X.509 standard.

1.2 Normative References

[FIPS186-3] *Digital Signature Standard (DSS)*, FIPS PUB 186-3, June 2009, http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf

[FIPS197] *Advanced Encryption Standard*, FIPS PUB 197, Nov 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[FIPS198-1] *The Keyed-Hash Message Authentication Code (HMAC)*, FIPS PUB 198-1, July 2008, http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

[IEEE1003-1] IEEE Std 1003.1, *Standard for information technology - portable operating system interface (POSIX). Shell and utilities*, 2004.

[ISO16609] ISO, *Banking -- Requirements for message authentication using symmetric techniques*, ISO 16609, 1991

[ISO9797-1] ISO/IEC, *Information technology -- Security techniques -- Message Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher*, ISO/IEC 9797-1, 1999.

[KMIP-Prof] OASIS Committee Draft 04, *Key Management Interoperability Protocol Profiles Version 1.0,* November 2009 http://docs.oasis-open.org/kmip/profiles/v1.0/cd04/kmip-profiles-1.0-cd-04.doc

[PKCS#1] RSA Laboratories, *PKCS #1 v2.1: RSA Cryptography Standard*, June 14, 2002. http://www.rsa.com/rsalabs/node.asp?id=2125

[PKCS#5] RSA Laboratories, *PKCS #5 v2.1: Password-Based Cryptography Standard*, October 5, 2006. http://www.rsa.com/rsalabs/node.asp?id=2127

[PKCS#7] RSA Laboratories, *PKCS#7 v1.5: Cryptographic Message Syntax Standard.* November 1, 1993*.* http://www.rsa.com/rsalabs/node.asp?id=2129

[PKCS#8] RSA Laboratories, *PKCS#8 v1.2: Private-Key Information Syntax Standard*, November 1, 1993. http://www.rsa.com/rsalabs/node.asp?id=2130

[PKCS#10] RSA Laboratories, *PKCS #10 v1.7: Certification Request Syntax Standard*, May 26, 2000. http://www.rsa.com/rsalabs/node.asp?id=2132

[RFC1319] B. Kaliski, *The MD2 Message-Digest Algorithm*, IETF RFC 1319, Apr 1992, http://www.ietf.org/rfc/rfc1319.txt

[RFC1320] R. Rivest, *The MD4 Message-Digest Algorithm*, IETF RFC 1320, Apr 1992, http://www.ietf.org/rfc/rfc1320.txt

[RFC1321] R. Rivest, *The MD5 Message-Digest Algorithm*, IETF RFC 1321, Apr 1992, http://www.ietf.org/rfc/rfc1321.txt

[RFC1421] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*¸IETF RFC 1421, Feb 1993, http://www.ietf.org/rfc/rfc1421.txt

[RFC1424] B. Kaliski, *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services*, IETF RFC 1424, February 1993. http://www.ietf.org/rfc/rfc1424.txt

[RFC2104] H. Krawczyk, M. Bellare, R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, IETF RFC 2104. Feb 1007, http://www.ietf.org/rfc/rfc2104.txt

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[RFC2898] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*, IETF RFC 2898, Sep 2000, http://www.ietf.org/rfc/rfc2898.txt

[RFC 3394] J. Schaad, R. Housley, *Advanced Encryption Standard (AES) Key Wrap Algorithm*, IETF RFC 3394, Sep 2002, http://www.ietf.org/rfc/rfc3394.txt

[RFC3447] J. Jonsson, B. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, IETF RFC 3447 Feb 2003, http://www.ietf.org/rfc/rfc3447.txt

[RFC3629] F. Yergeau, *UTF-8, a transformation format of ISO 10646*, IETF RFC 3629, Nov 2003, http://www.ietf.org/rfc/rfc3629.txt

[RFC3647] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, IETF RFC 3647, November 2003. http://www.ietf.org/rfc/rfc3647.txt

[RFC4210] C. Adams, S. Farrell, T. Kause and T. Mononen, *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*, IETF RFC 2510, September 2005. http://www.ietf.org/rfc/rfc4210.txt

[RFC4211] J. Schaad*, Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF),* IETF RFC 4211, Sep 2005, http://www.ietf.org/rfc/rfc4211.txt

[RFC4868] S. Kelly, S. Frankel, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*, IETF RFC 4868, May 2007, http://www.ietf.org/rfc/rfc4868.txt

[RFC4949] R. Shirey, *Internet Security Glossary, Version 2*, IETF RFC 4949, August 2007. http://www.ietf.org/rfc/rfc4949.txt

[RFC5272] J. Schaad and M. Meyers, *Certificate Management over CMS (CMC)*, IETF RFC 5272, June 2008. http://www.ietf.org/rfc/rfc5272.txt

[RFC5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk*, Internet X.509 Public Key Infrastructure Certificate*, IETF RFC 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt

[RFC5649] R. Housley, *Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm*, IETF RFC 5649, Aug 2009, http://www.ietf.org/rfc/rfc5649.txt

[SP800-38A] M. Dworkin, *Recommendation for Block Cipher Modes of Operation – Methods and Techniques*, NIST Special Publication 800-38A, Dec 2001, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[SP800-38B] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, NIST Special Publication 800-38B, May 2005, http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[SP800-38C] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*, NIST Special Publication 800-38C, May 2004, http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf

[SP800-38D] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication 800-38D, Nov 2007, http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[SP800-38E] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Block-Oriented Storage Devices*, NIST Special

Publication 800-38E, Aug 2009 (draft), http://csrc.nist.gov/publications/drafts/800-38E/draft-sp800-38E.pdf

[SP800-57-1] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendations for Key Management - Part 1: General (Revised),* NIST Special Publication 800-57 part 1, March 2007, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

[SP800-67] W. Barker, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, NIST Special Publication 800-67, Version 1.1, Revised 19 May 2008, http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf

[SP800-108] L. Chen, *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*, NIST Special Publication 800-108, October 2009, http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf

[X.509] International Telecommunication Union (ITU)–T, *X.509: Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks*, August 2005. http://www.itu.int/rec/T-REC-X.509-200508-I/en

[X9.24-1] ANSI, *X9.24 - Retail Financial Services Symmetric Key Management - Part 1: Using Symmetric Techniques*, 2004.

[X9.31] ANSI, *X9.31:Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, September 1998.

[X9.42] ANSI, *X9-42: Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography*, 2003.

[X9-57] ANSI, *X9-57: Public Key Cryptography for the Financial Services Industry: Certificate Management*, 1997.

[X9.62] ANSI, *X9-62: Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 2005.

[X9-63] ANSI, *X9-63: Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography*, 2001.

[X9-102] ANSI, *X9-102: Symmetric Key Cryptography for the Financial Services Industry - Wrapping of Keys and Associated Data*, 2008.

[X9 TR-31] ANSI, *X9 TR-31: Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms*, 2005.

1.3 Non-normative References

[KMIP-UG] OASIS Committee Draft 05, *Key Management Interoperability Protocol Usage Guide Version 1.0,* November 2009. http://docs.oasis-open.org/kmip/ug/v1.0/cd05/kmip-ug-1.0-cd-05.doc

[KMIP-UC] OASIS Committee Draft 05, *Key Management Interoperability Protocol Use Cases Version 1.0,* November 2009. http://docs.oasis-open.org/kmip/usecases/v1.0/cd05/kmip-usecases-1.0-cd-05.doc

[ISO/IEC 9945-2] The Open Group, *Regular Expressions, The Single UNIX Specification version 2*, 1997, ISO/IEC 9945-2:1993, http://www.opengroup.org/onlinepubs/007908799/xbd/re.html

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 of which they are composed. 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
- Byte String – sequence of unencoded byte values
- Date-Time – date and time, with a granularity of one second
- Interval – a length of time expressed in seconds

Deleted: interval

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

2.1 Base Objects

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

2.1.1 Attribute

An Attribute object is a structure (see Table 1) used for sending and receiving Managed Object attributes. The *Attribute Name* is a text-string that 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 Attribute Index is used to identify the particular instance. Attribute Indices SHALL start with 0. The Attribute Index of an attribute SHALL NOT change when other instances are added or deleted. For example, if a particular attribute has 4 instances with Attribute Indices 0, 1, 2 and 3, and the instance with Attribute Index 2 is deleted, then the Attribute Index of instance 3 is not changed. Attributes that have a single instance have an Attribute Index of 0, which is assumed if the Attribute Index is not specified. The *Attribute Value* is either a primitive data type or structured object, depending on the attribute.

Deleted: Table 1

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

Deleted: 3

Table 1: Attribute Object Structure

2.1.2 Credential

A *Credential* is a structure (see Table 2) used for client identification purposes and is not managed by the key management system (e.g., user id/password pairs, Kerberos tokens, etc). It MAY be used for authentication purposes as indicated in **[KMIP-Prof]**.

Deleted: Table 2

Object	Encoding	REQUIRED
Credential	Structure	
Credential Type	Enumeration, see 9.1.3.2.1	Yes
Credential Value	Byte String	Yes

Deleted: 9.1.3.2.1

Table 2: Credential Object Structure

2.1.3 Key Block

A *Key Block* object is a structure (see Table 3) used to encapsulate all of the information that is closely associated with a cryptographic key. It contains a Key Value of one of the following *Key Format Types*:

Deleted: Table 3

- *Raw* – This is a key that contains only 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 the RSAPrivateKey syntax and EncryptedPrivateKey.
- *X*.509 – This is an encoded object, expressed as a DER-encoded ASN.1 X.509 object.
- ECPrivateKey – This is an ASN.1 encoded elliptic curve private key.
- Several *Transparent Key* types – These are algorithm-specific structures containing defined values for the various key types, as defined in Section 2.1.7.
- *Extensions* – These are vendor-specific extensions to allow for proprietary or legacy key formats.

Deleted: 2.1.7

The Key Block MAY contain the Key Compression Type, which indicates the format of the elliptic curve public key. By default, the public key is uncompressed.

The Key Block also has the Cryptographic Algorithm and the Cryptographic Length of the key contained in the Key Value field. 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 SHALL contain a Key Wrapping Data structure if the key in the Key Value field is wrapped (i.e., encrypted, or MACed/signed, or both).

Object	Encoding	REQUIRED
Key Block	Structure	
Key Format Type	Enumeration, see 9.1.3.2.3	Yes
Key Compression Type	Enumeration, see 9.1.3.2.2	No
Key Value	Byte String: for wrapped Key Value; Structure: for plaintext Key Value, see 2.1.4	Yes
Cryptographic Algorithm	Enumeration, see 9.1.3.2.12	Yes, MAY be omitted only if this information is available from the Key Value. Does not apply to Secret Data or Opaque Objects. If present, the Cryptographic Length SHALL also be present.
Cryptographic Length	Integer	Yes, MAY be omitted only if this information is available from the Key Value. Does not apply to Secret Data or Opaque Objects. If present, the Cryptographic Algorithm SHALL also be present.
Key Wrapping Data	Structure, see 2.1.5	No, SHALL only be present if the key is wrapped.

Deleted: 9.1.3.2.3

Deleted: 9.1.3.2.2

Deleted: 2.1.4

Deleted: 9.1.3.2.12

Deleted: 2.1.5

Table 3: Key Block Object Structure

2.1.4 Key Value

The *Key Value* is used only inside a Key Block and is either a Byte String or a structure (see Table 4):

Deleted: Table 4

- The Key Value structure contains the key material, either as a byte string or as a Transparent Key structure (see Section 2.1.7), and OPTIONAL attribute information that is associated 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 possibly wrapped with) the key material itself.

Deleted: 2.1.7

- The Key Value Byte String is the wrapped TTLV-encoded (see Section 9.1) Key Value structure.

Deleted: 9.1

Object	Encoding	REQUIRED
Key Value	Structure	
Key Material	Byte String: for Raw, Opaque, PKCS1, PKCS8, ECPrivateKey, or Extension Key Format types; Structure: for Transparent, or Extension Key Format Types	Yes
Attribute	Attribute Object, see Section 2.1.1	No. MAY be repeated

Deleted: 2.1.1

Table 4: Key Value Object Structure

2.1.5 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 (see Table 5). It is only used inside a Key Block.

Deleted: Table 5

This structure contains fields for:

- A *Wrapping Method*, which indicates the method used to wrap the Key Value.
- *Encryption Key Information,* which contains the Unique Identifier (see 3.1) value of the encryption key and associated cryptographic parameters.
- *MAC/Signature Key Information,* which contains the Unique Identifier value of the MAC/signature key and associated cryptographic parameters.
- A *MAC/Signature,* which contains a MAC or signature of the Key Value.
- An *IV/Counter/Nonce,* if REQUIRED by the wrapping method.

Deleted: 3.1

If wrapping is used, then the whole Key Value structure is wrapped unless otherwise specified by the Wrapping Method. The algorithms used for wrapping are given by the Cryptographic Algorithm attributes of the encryption key and/or MAC/signature key; the block-cipher mode, padding method, and hashing algorithm used for wrapping are given by the Cryptographic Parameters in the Encryption Key Information and/or MAC/Signature Key Information, or, if not present, from the Cryptographic Parameters attribute of the respective key(s). At least one of the Encryption Key Information and the MAC/Signature Key Information SHALL be specified.

The following wrapping methods are currently defined:

- *Encrypt* only (i.e., encryption using a symmetric key or public key, or authenticated encryption algorithms that use a single key)
- *MAC/sign* only (i.e., either MACing the Key Value with a symmetric key, or signing the Key Value with a private key)
- *Encrypt then MAC/sign*
- *MAC/sign then encrypt*
- *TR-31*
- *Extensions*

Object	Encoding	REQUIRED
Key Wrapping Data	Structure	
Wrapping Method	Enumeration, see 9.1.3.2.4	Yes
Encryption Key Information	Structure, see below	No. Corresponds to the key that was used to encrypt the Key Value.
MAC/Signature Key Information	Structure, see below	No. Corresponds to the symmetric key used to MAC the Key Value or the private key used to sign the Key Value
MAC/Signature	Byte String	No
IV/Counter/Nonce	Byte String	No

Deleted: 9.1.3.2.4

Table 5: Key Wrapping Data Object Structure

The structures of the Encryption Key Information (see Table 6) and the MAC/Signature Key Information (see Table 7) are as follows:

Deleted: Table 6

Deleted: Table 7

Object	Encoding	REQUIRED
Encryption Key Information	Structure	
Unique Identifier	Text string, see **Error! Reference source not found.**	Yes
Cryptographic Parameters	Structure, see 3.6	No

Deleted: 3.1

Deleted: 3.6

Table 6: Encryption Key Information Object Structure

Object	Encoding	REQUIRED
MAC/Signature Key Information	Structure	
Unique Identifier	Text string, see **Error! Reference source not found.**	Yes. It SHALL be either the Unique Identifier of the Symmetric Key used to MAC, or of the Private Key (or its corresponding Public Key) used to sign.
Cryptographic Parameters	Structure, see 3.6	No

Deleted: 3.1

Deleted: 3.6

Table 7: MAC/Signature Key Information Object Structure

2.1.6 Key Wrapping Specification

This is a separate structure (see Table 8) that is defined for operations that provide the option to return wrapped keys. The *Key Wrapping Specification* SHALL be included inside the operation request if clients request the server to return a wrapped key. If Cryptographic Parameters are specified in the Encryption Key Information and/or the MAC/Signature Key Information of the Key Wrapping Specification, then the server SHALL verify that they match one of the instances of the Cryptographic Parameters attribute of the corresponding key. If Cryptographic Parameters are omitted, then the server SHALL use the

Deleted: Table 8

Cryptographic Parameters attribute with the lowest Attribute Index of the corresponding key. If the corresponding key does not have any Cryptographic Parameters attribute, or if no match is found, then an error is returned.

Comment: EBB: Which result reason would apply? In 11.11 there is a reason that says "Cryptographic Parameters associated with the object do not exist or do not match those provided in the Encryption Key Information and/or Signature Key Information". Is this the one to use? Note that the reason is in response to a Get request. I don't see the key wrapping spec used anywhere else.

Comment: MBJ proposed resolution: No change. The purpose of Section 11 is to gather all the error cases in one place, which also reduces the number of changes that need to be made if an error is introduced/modified/removed.

This structure contains:

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

Deleted: An

Deleted: A

Object	Encoding	REQUIRED
Key Wrapping Specification	Structure	
Wrapping Method	Enumeration, see 9.1.3.2.4	Yes
Encryption Key Information	Structure, see 2.1.5	No, SHALL be present if MAC/Signature Key Information is omitted
MAC/Signature Key Information	Structure, see 2.1.5	No, SHALL be present if Encryption Key Information is omitted
Attribute Name	Text String	No, MAY be repeated

Deleted: 9.1.3.2.4

Deleted: 2.1.5

Deleted: 2.1.5

Table 8: Key Wrapping Specification Object Structure

2.1.7 Transparent Key Structures

Transparent Key structures describe the necessary parameters to obtain the key material. They are used in the Key Value structure.

Deleted: key material in a form that is easily interpreted by all participants in the protocol

2.1.7.1 Transparent Symmetric Key

If the Key Format Type in the Key Block is *Transparent Symmetric Key*, then Key Material is a structure as shown in Table 9.

Deleted: Table 9

Object	Encoding	REQUIRED
Key Material	Structure	
Key	Byte String	Yes

Table 9: Key Material Object Structure for Transparent Symmetric Keys

2.1.7.2 Transparent DSA Private Key

If the Key Format Type in the Key Block is *Transparent DSA Private Key*, then Key Material is a structure as shown in Table 10.

Deleted: Table 10

Object	Encoding	REQUIRED
Key Material	Structure	
P	Big Integer	Yes
Q	Big Integer	Yes
G	Big Integer	Yes
X	Big Integer	Yes

Table 10: Key Material Object Structure for Transparent DSA Private Keys

P is the prime modulus. Q is the prime divisor of P-1. G is the generator. X is the private key (refer to NIST FIPS PUB 186-3).

2.1.7.3 Transparent DSA Public Key

If the Key Format Type in the Key Block is *Transparent DSA Public Key*, then Key Material is a structure as shown in Table 11.

Deleted: Table 11

Object	Encoding	REQUIRED
Key Material	Structure	
P	Big Integer	Yes
Q	Big Integer	Yes
G	Big Integer	Yes
Y	Big Integer	Yes

Table 11: Key Material Object Structure for Transparent DSA Public Keys

P is the prime modulus. Q is the prime divisor of P-1. G is the generator. Y is the public key (refer to NIST FIPS PUB 186-3).

2.1.7.4 Transparent RSA Private Key

If the Key Format Type in the Key Block is *Transparent RSA Private Key*, then Key Material is a structure as shown in Table 12.

Deleted: Table 12

Object	Encoding	REQUIRED
Key Material	Structure	
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

Table 12: Key Material Object Structure for Transparent RSA Private Keys

One of the following SHALL be present (refer to RSA PKCS#1):

- Private Exponent

- P and Q (the first two prime factors of Modulus)
- Prime Exponent P and Prime Exponent Q.

2.1.7.5 Transparent RSA Public Key

If the Key Format Type in the Key Block is *Transparent RSA Public Key*, then Key Material is a structure as shown in Table 13.

Deleted: Table 13

Object	Encoding	REQUIRED
Key Material	Structure	
Modulus	Big Integer	Yes
Public Exponent	Big Integer	Yes

Table 13: Key Material Object Structure for Transparent RSA Public Keys

2.1.7.6 Transparent DH Private Key

If the Key Format Type in the Key Block is *Transparent DH Private Key*, then Key Material is a structure as shown in Table 14.

Deleted: Table 14

Object	Encoding	REQUIRED
Key Material	Structure	
P	Big Integer	Yes
G	Big Integer	Yes
Q	Big Integer	No
J	Big Integer	No
X	Big Integer	Yes

Table 14: Key Material Object Structure for Transparent DH Private Keys

P is the prime, P = JQ + 1. G is the generator G^Q = 1 mod P. Q is the prime factor of P-1. J is the OPTIONAL cofactor. X is the private key (refer to ANSI X9.42).

2.1.7.7 Transparent DH Public Key

If the Key Format Type in the Key Block is *Transparent DH Public Key*, then Key Material is a structure as shown in Table 15.

Deleted: Table 15

Object	Encoding	REQUIRED
Key Material	Structure	
P	Big Integer	Yes
G	Big Integer	Yes
Q	Big Integer	No
J	Big Integer	No
Y	Big Integer	Yes

Table 15: Key Material Object Structure for Transparent DH Public Keys

P is the prime, P = JQ + 1. G is the generator G^Q = 1 mod P. Q is the prime factor of P-1. J is the OPTIONAL cofactor. Y is the public key (refer to ANSI X9.42).

2.1.7.8 Transparent ECDSA Private Key

If the Key Format Type in the Key Block is *Transparent ECDSA Private Key*, then Key Material is a structure as shown in Table 16.

Deleted: Table 16

Object	Encoding	REQUIRED
Key Material	Structure	
Recommended Curve	Enumeration, see 9.1.3.2.5	Yes
D	Big Integer	Yes

Deleted: 9.1.3.2.5

Table 16: Key Material Object Structure for Transparent ECDSA Private Keys

D is the private key (refer to NIST FIPS PUB 186-3).

2.1.7.9 Transparent ECDSA Public Key

If the Key Format Type in the Key Block is *Transparent ECDSA Public Key*, then Key Material is a structure as shown in Table 17.

Deleted: Table 17

Object	Encoding	REQUIRED
Key Material	Structure	
Recommended Curve	Enumeration, see 9.1.3.2.5	Yes
Q String	Byte String	Yes

Deleted: 9.1.3.2.5

Table 17: Key Material Object Structure for Transparent ECDSA Public Keys

Q String is the public key (refer to NIST FIPS PUB 186-3).

2.1.7.10 Transparent ECDH Private Key

If the Key Format Type in the Key Block is *Transparent ECDH Private Key*, then Key Material is a structure as shown in Table 18.

Deleted: Table 18

Object	Encoding	REQUIRED
Key Material	Structure	
Recommended Curve	Enumeration, see 9.1.3.2.5	Yes
D	Big Integer	Yes

Deleted: 9.1.3.2.5

Table 18: Key Material Object Structure for Transparent ECDH Private Keys

2.1.7.11 Transparent ECDH Public Key

If the Key Format Type in the Key Block is *Transparent ECDH Public Key*, then Key Material is a structure as shown in Table 19.

Deleted: Table 19

Object	Encoding	REQUIRED
Key Material	Structure	
Recommended Curve	Enumeration, see 9.1.3.2.5	Yes
Q String	Byte String	Yes

Deleted: 9.1.3.2.5

Table 19: Key Material Object Structure for Transparent ECDH Public Keys

Q String is the public key (refer to NIST FIPS PUB 186-3).

2.1.7.12 Transparent ECMQV Private Key

Comment: EBB: Is there a difference between an ECDH key and an ECMQV key, or is this a way to indicate the key agreement algorithm or to separate key usage?

Comment: MBJ: proposed resolution: No change. Similar to the ECDH/ECDSA resolution.

If the Key Format Type in the Key Block is *Transparent ECMQV Private Key*, then Key Material is a structure as shown in Table 20.

Deleted: Table 20

Object	Encoding	REQUIRED
Key Material	Structure	
Recommended Curve	Enumeration, see 9.1.3.2.5	Yes
D	Big Integer	Yes

Deleted: 9.1.3.2.5

Table 20: Key Material Object Structure for Transparent ECMQV Private Keys

2.1.7.13 Transparent ECMQV Public Key

If the Key Format Type in the Key Block is *Transparent ECMQV Public Key*, then Key Material is a structure as shown in Table 21.

Deleted: Table 21

Object	Encoding	REQUIRED
Key Material	Structure	
Recommended Curve	Enumeration, see 9.1.3.2.5	Yes
Q String	Byte String	Yes

Deleted: 9.1.3.2.5

Table 21: Key Material Object Structure for Transparent ECMQV Public Keys

2.1.8 Template-Attribute Structures

These structures are used in various operations to provide the desired attribute values and/or template names in the request and to return the actual attribute 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
Template-Attribute, Common Template-Attribute, Private Key Template-Attribute, Public Key Template-Attribute	Structure	
Name	Structure, see 3.2	No, MAY be repeated.
Attribute	Attribute Object, see 2.1.1	No, MAY be repeated

Deleted: 3.2

Deleted: 2.1.1

Table 22: Template-Attribute Object Structure

Name is the Name attribute of the Template object defined in Section 2.2.6.

Deleted: 2.2.6

2.2 Managed Objects

Managed Objects are objects that are the subjects of key management operations, which are described in Sections 4 and 5. *Managed Cryptographic Objects* are the subset of Managed Objects that contain cryptographic material (e.g. certificates, keys, and secret data).

Deleted: 4

Deleted: 5

2.2.1 Certificate

A Managed Cryptographic Object that is a digital certificate (e.g., an encoded X.509 certificate).

Object	Encoding	REQUIRED
Certificate	Structure	
Certificate Type	Enumeration, see 9.1.3.2.6	Yes
Certificate Value	Byte String	Yes

Deleted: 9.1.3.2.6

Table 23: Certificate Object Structure

2.2.2 Symmetric Key

A Managed Cryptographic Object that is a symmetric key.

Object	Encoding	REQUIRED
Symmetric Key	Structure	
Key Block	Structure, see 2.1.3	Yes

Deleted: 2.1.3

Table 24: Symmetric Key Object Structure

2.2.3 Public Key

A Managed Cryptographic Object that is the public portion of an asymmetric key pair. This is only a public key, not a certificate.

Object	Encoding	REQUIRED
Public Key	Structure	
Key Block	Structure, see 2.1.3	Yes

Deleted: 2.1.3

Table 25: Public Key Object Structure

2.2.4 Private Key

A Managed Cryptographic Object that is the private portion of an asymmetric key pair.

Object	Encoding	REQUIRED
Private Key	Structure	
Key Block	Structure, see 2.1.3	Yes

Deleted: 2.1.3

Table 26: Private Key Object Structure

2.2.5 Split Key

A Managed Cryptographic Object that 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 MAY then be distributed to several key holders, for additional security. The *Split Key Parts* field indicates the total number of parts, and the *Split Key Threshold* field indicates 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 SHALL be at least 1 and SHALL be less than or equal to Split Key Parts.

Object	Encoding	REQUIRED
Split Key	Structure	
Split Key Parts	Integer	Yes
Key Part Identifier	Integer	Yes
Split Key Threshold	Integer	Yes
Split Key Method	Enumeration, see 9.1.3.2.7	Yes
Prime Field Size	Big Integer	No, REQUIRED only if Split Key Method is Polynomial Sharing Prime Field.
Key Block	Structure, see 2.1.3	Yes

Deleted: 9.1.3.2.7

Deleted: 2.1.3

Table 27: Split Key Object Structure

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.

Let *L* be the minimum number of bits needed to represent all values of the secret.

- When the Split Key Method is XOR, then 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 XORing all of the parts.
- When the Split Key Method is Polynomial Sharing Prime Field, then 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}), then 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):

  $\mathrm{GF}(2^{16}) \approx \mathrm{GF}(2^8)\,[y]/(y^2+y+m)$, where *m* 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,

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

  An element of GF(2^8) is represented as a byte. Addition and subtraction in GF(2^8) is performed as a bit-wise XOR of the bytes. 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 bytes (*u, v*). The element *m* is given by

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

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

  Addition and subtraction in GF(2^{16}) both correspond to simply XORing the bytes. 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

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

2.2.6 Template

A *Template* is a named Managed Object containing the client-settable attributes of a Managed Cryptographic Object (i.e., 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 client-settable attributes specified in a Template except the Name attribute apply to the template object itself, but instead apply to any object created 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.

An attribute specified in a Template is applicable either to the Template itself or to objects created using the Template.

Attributes applicable to the Template itself are: Unique Identifier, Object Type, Name, Initial Date, Archive Date, and Last Change Date.

Attributes applicable to objects created using the Template are:

- Cryptographic Algorithm
- Cryptographic Length
- Cryptographic Domain Parameters
- Cryptographic Parameters
- Operation Policy Name
- Cryptographic Usage Mask
- Usage Limits
- Activation Date
- Process Start Date
- Protect Stop Date
- Deactivation Date
- Object Group
- Application Specific Information
- Contact Information
- Custom Attribute

Object	Encoding	REQUIRED
Template	Structure	
Attribute	Attribute Object, see 2.1.1	Yes. MAY be repeated.

Deleted: 2.1.1

Table 28: Template Object Structure

2.2.7 Secret Data

A Managed Cryptographic Object containing a shared secret value that is not a key or certificate (e.g., a password). The Key Block of the *Secret Data* object contains a Key Value of the Opaque type. The Key Value MAY be wrapped.

Object	Encoding	REQUIRED
Secret Data	Structure	
Secret Data Type	Enumeration, see 9.1.3.2.8	Yes
Key Block	Structure, see 2.1.3	Yes

Deleted: 9.1.3.2.8

Deleted: 2.1.3

Table 29: Secret Data Object Structure

2.2.8 Opaque Object

A Managed Object that the key management server is possibly not able to interpret. The context information for this object MAY be stored and retrieved using Custom Attributes.

Object	Encoding	REQUIRED
Opaque Object	Structure	
Opaque Data Type	Enumeration, see 9.1.3.2.9	Yes
Opaque Data Value	Byte String	Yes

Deleted: 9.1.3.2.9

Table 30: Opaque Object Structure

3 Attributes

The following subsections describe the attributes that are associated with Managed Objects. Attributes that an object MAY have multiple instances of are referred to as *multi-instance attributes*. Similarly, attributes which an object MAY only have at most one instance of are referred to as *single-instance attributes*. These attributes are able to be obtained by a client from the server using the Get Attribute operation. Some attributes are able to be set by the Add Attribute operation or updated by the Modify Attribute operation, and some are able to be deleted by the Delete Attribute operation if they no longer apply to the Managed Object. *Read-only attributes* are attributes that SHALL NOT be modified by either server or client, and that SHALL NOT be deleted by a client.

Formatted: Font: Italic

When attributes are returned by the server (e.g., via a Get Attributes operation), the attribute value returned MAY differ for different clients (e.g., the Cryptographic Usage Mask value MAY be different for different clients, depending on the policy of the server).

Deleted: attribute value

Deleted: depending on the client

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

Deleted: The

Deleted: contained in the first row of the Object

Deleted: column of the first table in each subsection

A server SHALL NOT delete attributes without receiving a request from a client until the object is destroyed. After an object is destroyed, the server MAY retain all, some or none of the object attributes, depending on the object type and server policy.

The second table in each subsection lists certain attribute characteristics (e.g., "SHALL always have a value"): Table 31 below explains the meaning of each characteristic that may appear in those tables. The server policy MAY further restrict these attribute characteristics.

Deleted: (see Table 31)

Deleted: .

Deleted: Table 31

Deleted: The "When implicitly set" characteristic indicates which operations (other than operations that manage attributes) are able to implicitly add to or modify the attribute of the object, which MAY be object(s) on which the operation is performed or object(s) created as a result of the operation. Implicit attribute changes MAY occur even if the attribute is not specified in the operation request itself.

SHALL always have a value	All Managed Objects that are of the Object Types for which this attribute applies, SHALL always have this attribute set once the object has been created or registered, up until the object has been destroyed.
Initially set by	Who is permitted to initially set the value of the attribute (if the attribute has never been set, or if all the attribute values have been deleted)?
Modifiable by server	Is the server allowed to change an existing value of the attribute without receiving a request from a client?
Modifiable by client	Is the client able to change an existing value of the attribute value once it has been set?
Deletable by client	Is the client able to delete an instance of the attribute?
Multiple instances permitted	Are multiple instances of the attribute permitted?
When implicitly set	Which operations MAY cause this attribute to be set even if the attribute is not specified in the operation request itself?
Applies to Object Types	Which Managed Objects MAY have this attribute set?

Table 31: Attribute Rules

3.1 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 in order to allow for a key management domain export of such objects. This attribute SHALL be assigned by the key management system at creation or registration time, and then SHALL NOT be changed or deleted before the object is destroyed.

Object	Encoding	
Unique Identifier	Text String	

Table 32: Unique Identifier Attribute

Deleted: modify

Deleted: modify

Deleted: without an explicit request from a client

Deleted: ,

Comment: EBB: Does this mean, for example, that the identifier of a key is never destroyed when the key is destroyed? Would this result in a huge number of identifiers lying around that are no longer associated with any object any more? Isn't this a matter of server policy (in the case where a key is destroyed)?

Comment: MBJ: proposed resolution: Added clarification to Section 3.

Deleted: by any entity at any time.

SHALL 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

Table 33: Unique Identifier Attribute Rules

3.2 Name

The *Name* attribute is a structure (see Table 34) used to identify and locate the object. This attribute is assigned by the client, and the *Name Value* is intended to be in a form that humans are able to interpret. The key management system MAY specify rules by which the client creates valid names. Clients are informed of such rules by a mechanism that is not specified by this standard. Names SHALL be unique within a given key management domain, but are not REQUIRED to be globally unique.

Deleted: Table 34

Deleted: ,

Formatted: Font: Italic

Object	Encoding	REQUIRED
Name	Structure	
Name Value	Text String	Yes
Name Type	Enumeration, see 9.1.3.2.10	Yes

Deleted: 9.1.3.2.10

Table 34: Name Attribute Structure

SHALL 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

Comment: EBB: Is the client notified?

Comment: MBJ: proposed resolution: No change. Maybe if the client supports server-to-client operations, otherwise no

Table 35: Name Attribute Rules

3.3 Object Type

The *Object Type* of a Managed Object (e.g., public key, private key, symmetric key, etc) SHALL be set by the server when the object is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

Deleted: . This attribute

Object	Encoding	
Object Type	Enumeration, see 9.1.3.2.11	

Deleted: 9.1.3.2.11

Table 36: Object Type Attribute

SHALL 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

Table 37: Object Type Attribute Rules

3.4 Cryptographic Algorithm

The *Cryptographic Algorithm* used by the object (e.g., RSA, DSA, DES, 3DES, AES, etc). This attribute SHALL be set by the server when the object is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

Object	Encoding	
Cryptographic Algorithm	Enumeration, see 9.1.3.2.12	

Deleted: 9.1.3.2.12

Table 38: Cryptographic Algorithm Attribute

SHALL 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	Keys, Certificates, Templates

Table 39: Cryptographic Algorithm Attribute Rules

3.5 Cryptographic Length

Cryptographic Length is the length in bits of the clear-text cryptographic key material of the Managed Cryptographic Object. This attribute SHALL be set by the server when the object is created or registered, and then SHALL NOT be changed or deleted before the object is destroyed.

Object	Encoding	
Cryptographic Length	Integer	

Table 40: Cryptographic Length Attribute

SHALL 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	Keys ,Certificates, Templates

Table 41: Cryptographic Length Attribute Rules

3.6 Cryptographic Parameters

The *Cryptographic Parameters* attribute is a structure (see Table 42) 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 pertain only to certain types of Managed Cryptographic Objects.

Object	Encoding	REQUIRED
Cryptographic Parameters	Structure	
Block Cipher Mode	Enumeration, see 9.1.3.2.13	No
Padding Method	Enumeration, see 9.1.3.2.14	No
Hashing Algorithm	Enumeration, see 9.1.3.2.15	No
Role Type	Enumeration, see 9.1.3.2.16	No

Table 42: Cryptographic Parameters Attribute Structure

SHALL 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	Keys ,Certificates, Templates

Table 43: Cryptographic Parameters Attribute Rules

Role Type definitions match those defined in ANSI X9 TR-31 **[X9 TR-31]** and are defined in Table 44.

Deleted: Table 42

Deleted: It is possible that s

Deleted: pertain

Deleted: 9.1.3.2.13

Deleted: 9.1.3.2.14

Deleted: 9.1.3.2.15

Deleted: 9.1.3.2.16

Comment: EBB: Is this really a good idea? Once these values are determined for a given key, would it be a security risk to allow them to be changed (e.g., to change the role of a key from a DEK to a KEK), Someone needs to look at this carefully. Or is it the server's responsibility to check this? Where do we provide guidance/assumptions about what a server needs to do?

Comment: EBB: This may have some of the same problems as stated in the previous comment (e.g., a key used as both a DEK and a KEK).

Comment: MBJ: proposed resolution: added clarification to beginning of Section 3. The Cryptographic Usage Mask indicates what the key can be used for.

Comment: EBB: I think this table is in 9.1.3.2.16, so is not needed here. It should be in only one place.Duplicatio

Comment: MBJ: proposed resolution: No change. The table in 9.1.3.2.16 does not contain the definitions

Deleted: Table 44

BDK	Base Derivation Key (ANSI X9.24 DUKPT key derivation)
CVK	Card Verification Key (CVV/signature strip number validation)
DEK	Data Encryption Key (General Data Encryption)
MKAC	EMV/chip card Master Key: Application Cryptograms
MKSMC	EMV/chip card Master Key: Secure Messaging for Confidentiality
MKSMI	EMV/chip card Master Key: Secure Messaging for Integrity
MKDAC	EMV/chip card Master Key: Data Authentication Code
MKDN	EMV/chip card Master Key: Dynamic Numbers
MKCP	EMV/chip card Master Key: Card Personalization
MKOTH	EMV/chip card Master Key: Other
KEK	Key Encryption or Wrapping Key
MAC16609	ISO16609 MAC Algorithm 1
MAC97971	ISO9797-1 MAC Algorithm 1
MAC97972	ISO9797-1 MAC Algorithm 2
MAC97973	ISO9797-1 MAC Algorithm 3 (Note this is commonly known as X9.19 Retail MAC)
MAC97974	ISO9797-1 MAC Algorithm 4
MAC97975	ISO9797-1 MAC Algorithm 5
ZPK	PIN Block Encryption Key
PVKIBM	PIN Verification Key, IBM 3624 Algorithm
PVKPVV	PIN Verification Key, VISA PVV Algorithm
PVKOTH	PIN Verification Key, Other Algorithm

Table 44: Role Types

Accredited Standards Committee X9, Inc. - Financial Industry Standards (www.x9.org) contributed to Table 44. Key role names and descriptions are derived from material in the Accredited Standards Committee X9, Inc's Technical Report "TR-31 2005 Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms" and used with the permission of Accredited Standards Committee X9, Inc. in an effort to improve interoperability between X9 standards and OASIS KMIP. The complete ANSI X9 TR-31 is available at www.x9.org.

Deleted: Table 44

Comment: EBB: don't need this

Comment: MBJ: proposed resolution: No change, this text was a requirement in order to use the table.

3.7 Cryptographic Domain Parameters

The *Cryptographic Domain Parameters* attribute is a structure (see Table 45) that contains a set of OPTIONAL fields that MAY need to be specified in the Create Key Pair Request Payload. Specific fields MAY only pertain to certain types of Managed Cryptographic Objects.

Deleted: Table 45

For DSA, the domain parameter Qlength correponds to the length of the parameter Q in bits. The length of P needs to be specified separately by setting the Cryptographic Length attribute.

Comment: EBB: Is the assumption here that "Qlength" is used for DSA, and "Recommended Curve" is used for ECDSA. Note that "Qlength" could also be used for FF DH and MQV, and that "Recommended Curve" could also be used for ECDH and ECMQV.

Comment: MBJ: proposed resolution: TBD

Object	Encoding	Required
Cryptographic Domain Parameters	Structure	Yes
Qlength	Integer	No
Recommended Curve	Enumeration, see 9.1.3.2.5	No

Table 45: Cryptographic Domain Parameters Attribute Structure

Shall always have a value	No
Initially set by	Client
Modifiable by server	No
Modifiable by client	No
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Re-key
Applies to Object Types	Asymmetric Keys, Templates

Table 46: Cryptographic Domain Parameters Attribute Rules

3.8 Certificate Type

The type of a certificate (e.g., X.509, PGP, etc). The *Certificate Type* value SHALL be set by the server when the certificate is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

Object	Encoding	
Certificate Type	Enumeration, see 9.1.3.2.6	

Deleted: 9.1.3.2.6

Table 47: Certificate Type Attribute

SHALL 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	Register, Certify, Re-certify
Applies to Object Types	Certificates

Table 48: Certificate Type Attribute Rules

3.9 Certificate Identifier

The *Certificate Identifier* attribute is a structure (see Table 49) used to provide the identification of a certificate, containing the Issuer Distinguished Name (i.e., from the Issuer field of the certificate) and the Certificate Serial Number (i.e., from the Serial Number field of the certificate). The Certificate Identifier

Deleted: Table 49

Deleted: is

Deleted: value

SHALL be set by the server when the certificate is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

Object	Encoding	REQUIRED
Certificate Identifier	Structure	
Issuer	Text String	Yes
Serial Number	Text String	Yes (for X.509 certificates) / No (for PGP certificates since they do not contain a serial number)

Table 49: Certificate Identifier Attribute Structure

SHALL 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	Register, Certify, Re-certify
Applies to Object Types	Certificates

Table 50: Certificate Identifier Attribute Rules

3.10 Certificate Subject

The *Certificate Subject* attribute is a structure (see Table 51) used to identify the subject of a certificate, containing the Subject Distinguished Name (i.e., from the Subject field of the certificate). It MAY include one or more alternative names (e.g., email address, IP address, DNS name) for the subject of the certificate (i.e., from the Subject Alternative Name extension within the certificate). These values SHALL be set by the server based on the information it extracts from the certificate that is created (as a result of a Certify or a Re-certify operation) or registered (as part of a Register operation) and SHALL NOT be changed or deleted before the object is destroyed.

Deleted: Table 51

Deleted: during the lifespan of the certificate

If the Subject Alternative Name extension is included in the certificate and is marked *CRITICAL* (i.e., within the certificate itself), then it is possible to issue an X.509 certificate where the subject field is left blank. Therefore an empty string is an acceptable value for the Certificate Subject Distinguished Name.

Object	Encoding	REQUIRED
Certificate Subject	Structure	
Certificate Subject Distinguished Name	Text String	Yes, but MAY be the empty string
Certificate Subject Alternative Name	Text String	No, MAY be repeated

Table 51: Certificate Subject Attribute Structure

SHALL 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	Register, Certify, Re-certify
Applies to Object Types	Certificates

Table 52: Certificate Subject Attribute Rules

3.11 Certificate Issuer

The *Certificate Issuer* attribute is a structure (see Table 54) used to identify the issuer of a certificate, containing the Issuer Distinguished Name (i.e., from the Issuer field of the certificate). It MAY include one or more alternative names (e.g., email address, IP address, DNS name) for the issuer of the certificate (i.e., from the Issuer Alternative Name extension within the certificate). The server SHALL set these values based on the information it extracts from a certificate that is created as a result of a Certify or a Re-certify operation or is sent as part of a Register operation. These values SHALL NOT be changed or deleted before the object is destroyed.

Deleted: Table 54

Deleted: during the lifespan of the certificate

Object	Encoding	REQUIRED
Certificate Issuer	Structure	
Certificate Issuer Distinguished Name	Text String	Yes
Certificate Issuer Alternative Name	Text String	No, MAY be repeated

Comment: EBB: Can this be empty, as is the case for the certificate subject distinguished name in 3.10?

Comment: MBJ: proposed resolution: No change.

Table 53: Certificate Issuer Attribute Structure

SHALL 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	Register, Certify, Re-certify
Applies to Object Types	Certificates

Table 54: Certificate Issuer Attribute Rules

3.12 Digest

The *Digest* attribute is a structure (see Table 55) that contains the digest value of the key or secret data (i.e., digest of the Key Material), certificate (i.e., digest of the Certificate Value), or opaque object (i.e., digest of the Opaque Data Value). Multiple digests MAY be calculated using different algorithms. The mandatory digest SHALL be computed with the SHA-256 hashing algorithm; the server MAY store additional digests using the algorithms listed in Section 9.1.3.2.15. The digest(s) are static and SHALL be generated by the server when the object is created or registered.

Deleted: Table 55

Deleted: 9.1.3.2.15

Object	Encoding	REQUIRED
Digest	Structure	
Hashing Algorithm	Enumeration, see 9.1.3.2.15	Yes
Digest Value	Byte String	Yes

Deleted: 9.1.3.2.15

Table 55: Digest Attribute Structure

SHALL always have a value	Yes
Initially set by	Server
Modifiable by server	No
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, Opaque Objects

Deleted: Yes

Table 56: Digest Attribute Rules

3.13 Operation Policy Name

An operation policy controls what entities MAY perform which key management operations on the object. The content of the *Operation Policy Name* attribute is the name of a policy object known to the key management system and, therefore, is server dependent. The named policy objects are created and managed using mechanisms outside the scope of the protocol. The policies determine what entities MAY perform specified operations on the object, and which of the object's attributes MAY be modified or deleted. The Operation Policy Name attribute SHOULD be set when operations that result in a new Managed Object on the server are executed. It is set either explicitly or via some default set by the server, which then applies the named policy to all subsequent operations on the object.

Comment: EBB: how?

Comment: MBJ: explicitly set by Add Attribute, or by specifying the attribute in Register, Create etc operations

Object	Encoding	
Operation Policy Name	Text String	

Table 57: Operation Policy Name Attribute

SHALL 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

Comment: EBB: There seems to be an assumption here about the conditions under which this could be done. Again, is there a discussion anywhere about what the server's responsibilities?

Comment: MBJ: proposed resolution: addressed by additional clarifications in Section 3

Table 58: Operation Policy Name Attribute Rules

3.13.1 Operations outside of operation policy control

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

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

3.13.2 Default Operation Policy

A key management system implementation SHALL implement at least one named operation policy, which is used for objects when the *Operation Policy* attribute is not specified by the Client in operations that result in a new Managed Object on the server, 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. For the profiles defined in **[KMIP-Prof]**, the creator SHALL be as defined in **[KMIP-Prof]**.

Formatted: Space Before: 6 pt

Deleted: a

Deleted: *Create* or *Register* operation

3.13.2.1 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

Table 59: Default Operation Policy for Secret Objects

3.13.2.2 Default Operation Policy for Certificates and Public Key Objects

Deleted: For mandatory profiles, the creator SHALL be the transport-layer identification (see **[KMIP-Prof]**) provided at the Create or Register operation time.¶

Formatted: Bullets and Numbering

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

Comment: EBB: why only to the creator (i.e., the CA)?

Comment: EBB: why only to the creator (i.e., the CA)?

Comment: EBB: why only to the creator (i.e., the CA)?

Comment: MBJ: proposed resolution: No change. Only creator is defined at the moment.

Table 60: Default Operation Policy for Certificates and Public Key Objects

Default Operation Policy for Template Objects

Formatted: Bullets and Numbering

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

Deleted: *Create*

Comment: EBB: This is difficult to parse. An example might help. Of course, part of the problem is that I just don't understand the use of templates. Could the user guide explain their use (e.g., provide an example)?

Comment: MBJ: proposed resolution: No change. Possibly UG.

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

Table 61: Default Operation Policy for Private Template Objects

In addition to private template objects (which are controlled by the above policy, and which MAY be created by clients or the server), publicly known and usable templates MAY be created and managed by the server, with a default policy different from private 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

Table 62: Default Operation Policy for Public Template Objects

3.14 Cryptographic Usage Mask

The *Cryptographic Usage Mask* defines the cryptographic usage of a key. This is a bit mask that indicates to the client which cryptographic functions MAY be performed using the key, and which ones SHALL NOT be performed.

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

- CRL Sign
- Generate Cryptogram
- Validate Cryptogram
- Translate Encrypt
- Translate Decrypt
- Translate Wrap
- Translate Unwrap

This list takes into consideration values that MAY appear in the Key Usage extension in an X.509 certificate. However, the list does not consider the additional usages that 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 (Non Repudiation)
keyEncipherment	Wrap Key and Unwrap Key
dataEncipherment	Encrypt and Decrypt
keyAgreement	Key Agreement
keyCertSign	Certificate Sign
cRLSign	CRL Sign
encipherOnly	Encrypt
decipherOnly	Decrypt

Comment: EBB: or? signing is done with the private key, whereas verification is done with the public key

Comment: MBJ: proposed resolution: TBD.

Table 63: X.509 Key Usage to Cryptographic Usage Mask Mapping

Object	Encoding	
Cryptographic Usage Mask	Integer	

Table 64: Cryptographic Usage Mask Attribute

SHALL 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, Templates

Comment: EBB: why?

Comment: MBJ: proposed resolution: No change. Crypto Usage Mask not an immutable object.

Table 65: Cryptographic Usage Mask Attribute Rules

3.15 Lease Time

The *Lease Time* attribute defines a time interval for a Managed Cryptographic Object beyond which the client SHALL NOT use the object without obtaining another lease. This attribute always holds the initial length of time allowed for a lease, and not the actual remaining time. Once its lease expires, the client is only able to renew the lease by calling Obtain Lease. A server SHALL store in this attribute the maximum Lease Time it is able to serve and a client obtains the lease time (with Obtain Lease) that is less than or equal to the maximum Lease Time. This attribute is read-only for clients. It SHALL be modified by the server only.

Deleted: value of

Deleted: the

Deleted: then

Comment: EBB: under what conditions? a compromise, for example?

Comment: MBJ: proposed resolution: No change. This could be a server policy change

Object	Encoding	
Lease Time	Interval	

Table 66: Lease Time Attribute

SHALL 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 Cryptographic Objects

Table 67: Lease Time Attribute Rules

3.16 Usage Limits

The *Usage Limits* attribute is a mechanism for limiting the usage of a Managed Cryptographic Object. It only applies to Managed Cryptographic Objects that are able to be used for applying cryptographic protection and it SHALL only reflect their usage for applying that protection (e.g., encryption, signing, etc.). This attribute does not necessarily exist for all Managed Cryptographic Objects, since some objects are able to be used without limit for cryptographically protecting data, depending on client/server policies. Usage for processing cryptographically-protected data (e.g., decryption, verification, etc.) is not limited.

The Usage Limits attribute has four fields for two different types of limits, bytes and objects. Exactly one of these two types SHALL be present when the Usage Limits attribute is used. These fields 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 SHALL NOT be changed once the object begins to be used for applying cryptographic protection.
- *Usage Limits Byte Count* – the currently remaining number of bytes allowed to be protected by the object.
- *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 SHALL NOT be changed once the object begins to be used for applying cryptographic protection.
- *Usage Limits Object Count* – the currently remaining number of objects allowed to be protected by the object.

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

Object	Encoding	REQUIRED
Usage Limits	Structure	
Usage Limits Total Bytes	Big Integer	No. SHALL be present if Usage Limits Byte Count is present
Usage Limits Byte Count	Big Integer	No. SHALL be present if Usage Limits Object Count is not present
Usage Limits Total Objects	Big Integer	No. SHALL be present if Usage Limits Object Count is present
Usage Limits Object Count	Big Integer	No. SHALL be present if Usage Limits Byte Count is not present

Table 68: Usage Limits Attribute Structure

SHALL always have a value	No
Initially set by	Server (Total and/or Count) or Client (Total only)
Modifiable by server	Yes
Modifiable by client	Yes (Total only, as long as Get Usage Allocation has not been performed)
Deletable by client	Yes, as long as Get Usage Allocation has not been performed
Multiple instances permitted	No
When implicitly set	Create, Create Key Pair, Register, Derive Key, Re-key, Get Usage Allocation
Applies to Object Types	Keys, Templates

Table 69: Usage Limits Attribute Rules

3.17 State

This attribute is an indication of the *State* of an object as known to the key management server. The State SHALL NOT be changed by using the Modify Attribute operation on this attribute. The state SHALL only be changed by the server as a part of other operations or other server processes. An object SHALL be in one of the following states at any given time. (Note: These states correspond to those described in NIST Special Publication 800-57 **[SP800-57-1]**).

- *Pre-Active*: The object exists but is not yet usable for any cryptographic purpose.
- *Active*: The object MAY be used for all cryptographic purposes that are allowed by its Cryptographic Usage Mask attribute and, if applicable, by its Process Start Date (see 3.20) and Protect Stop Date (see 3.21) attributes.
- *Deactivated*: The object SHALL NOT be used for applying cryptographic protection (e.g., encryption or signing), but, if permitted by the Cryptographic Usage Mask attribute, then the object MAY be used to process cryptographically-protected information (e.g., decryption or verification), but only under extraordinary circumstances and when special permission is granted.
- *Compromised*: It is possible that the object has been compromised, and SHOULD only be used to process cryptographically-protected information in a client that is trusted to use managed objects that have been compromised.
- *Destroyed*: The object is no longer usable for any purpose.

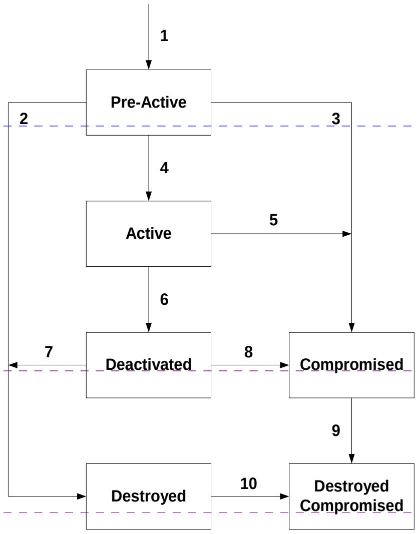

Figure 1: Cryptographic Object States and Transitions

Deleted: 3.20

Deleted: 3.21

Comment: EBB: This bothers me. A symmetric key could be used for encryption for a predetermined amount of time, but it's OK to use it for decryption for considerably longer. Is this considered to be an extraordinary circumstance? This could very well be the case for stored data.

Comment: MBJ: proposed resolution: No change. Process Start Date and Protect Stop Date deal with this use-case. The key is Active, but might only be used for one of the functions at a time (or both), depending on the values of these attributes.

Deleted: handle compromised cryptographic objects

- *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 the Pre-Active state is caused 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, then the state immediately transitions from Pre-Active to Active. In this case, the server SHALL set the Activation Date attribute to the time when the operation is received, or fail the request attempting to create or register the object, depending on server policy. If the operation contains an Activation Date attribute that is in the future, or contains no Activation Date, then the Cryptographic Object is initialized in the key management system in the Pre-Active state.
2. The transition from Pre-Active to Destroyed is caused by a client issuing a Destroy operation. The server destroys the object when (and if) server policy dictates.
3. The transition from Pre-Active to Compromised is caused by a client issuing a Revoke operation with a Revocation Reason of Compromised.
4. The transition from Pre-Active to Active SHALL occur in one of three ways:
    - The Activation Date is reached.
    - 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 SHALL either set the Activation Date attribute to the date in the past or the current date, or fail the operation, depending on server policy.
    - A client issues an Activate operation on the object. The server SHALL set the Activation Date to the time the Activate operation is received.
5. The transition from Active to Compromised is caused by a client issuing a Revoke operation with a Revocation Reason of Compromised.
6. The transition from Active to Deactivated SHALL occur in one of three ways:
    - The object's Deactivation Date is reached.
    - 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 SHALL either set the Deactivation Date attribute to the date in the past or the current date, or fail the operation, depending on server policy.
7. The transition from Deactivated to Destroyed is caused by a client issuing a Destroy operation, or by a server, both in accordance with server policy. The server destroys the object when (and if) server policy dictates.
8. The transition from Deactivated to Compromised is caused by a client issuing a Revoke operation with a Revocation Reason of Compromised.
9. The transition from Compromised to Destroyed Compromised is caused by a client issuing a Destroy operation, or by a server, both in accordance with server policy. The server destroys the object when (and if) server policy dictates.
10. The transition from Destroyed to Destroyed Compromised is caused by a client issuing a *R*evoke operation with a Revocation Reason of Compromised.

Only the transitions described above are permitted.

Object	Encoding	
State	Enumeration, see 9.1.3.2.17	

Deleted: object has an Activation Date in the future. At the time that the

Deleted: , the server changes the state to Active

Comment: EBB: Re setting it to a date in the past: the key could have have been used since then, so this change to the past date could be a problem.

Comment: MBJ: proposed resolution: No change. If the date in the Modify Attribute request is only "slightly" in the past (e.g. due to small differences in the client and server clocks), the server might accept the date in the past. If the date is too far in the past, the server can fail the operation due to the problem EBB mentioned. The behavior is dictated by server policy. (the current wording is due to disallowing of the mutations of attributes by the server)

Comment: EBB: the server has to have a policy about a client requesting destruction.

Comment: MBJ: proposed resolution: No change. See Destroy in Section 4, UG.

Deleted: 9.1.3.2.17

Table 70: State Attribute

SHALL always have a value	Yes
Initially set by	Server
Modifiable by server	Yes
Modifiable by client	No, but only by the server in response to certain requests (see above)
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

Table 71: State Attribute Rules

3.18 Initial Date

The *Initial Date* is the date and time when the Managed Object was first created or registered at the server. This time corresponds to state transition 1 (see Section 3.17). This attribute SHALL be set by the server when the object is created or registered, and then SHALL NOT be changed or deleted before the object is destroyed. This attribute is also set for non-cryptographic objects (e.g., templates) when they are first registered with the server.

Deleted: 3.17

Object	Encoding	
Initial Date	Date-Time	

Table 72: Initial Date Attribute

SHALL 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

Table 73: Initial Date Attribute Rules

3.19 Activation Date

This is the date and time when the Managed Cryptographic Object MAY begin to be used. This time corresponds to state transition 4 (see Section 3.17). The object SHALL NOT be used for any cryptographic purpose before the *Activation Date* has been reached. Once the state transition from Pre-Active has occurred, then this attribute SHALL NOT be changed or deleted before the object is destroyed

Deleted: 3.17

Deleted: modified

Deleted: by the server or client

Object	Encoding	
Activation Date	Date-Time	

Table 74: Activation Date Attribute

SHALL always have a value	No
Initially set by	Server or Client
Modifiable by server	Yes, only while in Pre-Active state
Modifiable by client	Yes, only while in Pre-Active state
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 Cryptographic Objects, Templates

Table 75: Activation Date Attribute Rules

3.20 Process Start Date

This is the date and time when a Managed Symmetric Key Object MAY begin to be used to process cryptographically-protected information (e.g., decryption or unwrapping), depending on the value of its Cryptographic Usage Mask attribute. The object SHALL NOT be used for these cryptographic purposes before the *Process Start Date* has been reached. This value MAY be equal to or later than, but SHALL NOT precede, the Activation Date. Once the Process Start Date has occurred, then this attribute SHALL NOT be changed or deleted before the object is destroyed.

Deleted: modified

Deleted: by the server or the client

Object	Encoding	
Process Start Date	Date-Time	

Table 76: Process Start Date Attribute

SHALL always have a value	No
Initially set by	Server or Client
Modifiable by server	Yes, only while in Pre-Active or Active state and as long as the Process Start Date has been not reached.
Modifiable by client	Yes, only while in Pre-Active or Active state and as long as the Process Start Date has been not reached.
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Create, Register, Derive Key, Re-key
Applies to Object Types	Symmetric Keys, Split Keys of symmetric keys, Templates

Table 77: Process Start Date Attribute Rules

3.21 Protect Stop Date

This is the date and time when a Managed Symmetric Key Object SHALL NOT be used for applying cryptographic protection (e.g., encryption or wrapping), depending on the value of its Cryptographic Usage Mask attribute. This value MAY be equal to or earlier than, but SHALL NOT be later than the Deactivation Date. Once the *Protect Stop Date* has occurred, then this attribute SHALL NOT be changed or deleted before the object is destroyed.

Deleted: modified by the server or the client

Object	Encoding	
Protect Stop Date	Date-Time	

Table 78: Protect Stop Date Attribute

SHALL always have a value	No
Initially set by	Server or Client
Modifiable by server	Yes, only while in Pre-Active or Active state and as long as the Protect Stop Date has not been reached.
Modifiable by client	Yes, only while in Pre-Active or Active state and as long as the Protect Stop Date has not been reached.
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Create, Register, Derive Key, Re-key
Applies to Object Types	Symmetric Keys, Split Keys of symmetric keys, Templates

Table 79: Protect Stop Date Attribute Rules

3.22 Deactivation Date

The *Deactivation Date* is the date and time when the Managed Cryptographic Object SHALL NOT be used for any purpose, except for decryption, signature verification, or unwrapping, but only under extraordinary circumstances and only when special permission is granted. This time corresponds to state transition 6 (see Section 3.17). This attribute SHALL NOT be changed or deleted before the object is destroyed, unless the object is in the Pre-Active or Active state.

Deleted: 3.17

Deleted: Once

Deleted: this transition has occurred, then this

Deleted: modified

Deleted: by the server or client

Object	Encoding	
Deactivation Date	Date-Time	

Table 80: Deactivation Date Attribute

SHALL always have a value	No
Initially set by	Server or Client
Modifiable by server	Yes, only while in Pre-Active or Active state
Modifiable by client	Yes, only while in Pre-Active or Active state
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, Templates

Table 81: Deactivation Date Attribute Rules

3.23 Destroy Date

The *Destroy Date* is the date and time when the Managed Object was destroyed. This time corresponds to state transitions 2, 7, or 9 (see Section 3.17). This value is set by the server when the object is destroyed due to the reception of a Destroy operation, or due to server policy or out-of-band administrative action.

Deleted: 3.17

Object	Encoding	
Destroy Date	Date-Time	

Table 82: Destroy Date Attribute

SHALL 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 Cryptographic Objects, Opaque Objects

Table 83: Destroy Date Attribute Rules

3.24 Compromise Occurrence Date

The *Compromise Occurrence Date* is 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, then this value SHOULD be set to the Initial Date for the object.

Object	Encoding	
Compromise Occurrence Date	Date-Time	

Table 84: Compromise Occurrence Date Attribute

SHALL 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	All Cryptographic Objects, Opaque Object

Table 85: Compromise Occurrence Date Attribute Rules

3.25 Compromise Date

The *Compromise Date* is the date and time when the Managed Cryptographic Object entered into the compromised state. This time corresponds to state transitions 3, 5, 8, or 10 (see Section 3.17). This time indicates 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, or due to server policy or out-of-band administrative action.

Deleted: 3.17

Object	Encoding	
Compromise Date	Date-Time	

Table 86: Compromise Date Attribute

SHALL 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	All Cryptographic Objects, Opaque Object

Table 87: Compromise Date Attribute Rules

3.26 Revocation Reason

The *Revocation Reason* attribute is a structure (see Table 88) used to indicate 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 part of the Revoke Operation.

Deleted: Table 88

The *Revocation Message* is an OPTIONAL field that is used exclusively for audit trail/logging purposes and MAY contain additional information about why the object was revoked (e.g., "Laptop stolen", or "Machine decommissioned").

Object	Encoding	REQUIRED
Revocation Reason	Structure	
Revocation Reason Code	Enumeration, see 9.1.3.2.18	Yes
Revocation Message	Text String	No

Deleted: 9.1.3.2.18

Table 88: Revocation Reason Attribute Structure

SHALL 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, Opaque Object

Table 89: Revocation Reason Attribute Rules

3.27 Archive Date

The *Archive Date* is the date and time when the Managed Object was placed in archival storage. This value is set by the server as a part of the Archive operation. The server SHALL delete this attribute whenever a Recover operation is performed.

Deleted: is

Deleted: is deleted

Object	Encoding	
Archive Date	Date-Time	

Table 90: Archive Date Attribute

SHALL 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	Archive
Applies to Object Types	All Objects

Deleted: Yes

Table 91: Archive Date Attribute Rules

3.28 Object Group

An object MAY be part of a group of objects. An object MAY belong to more than one group of objects. To assign an object to a group of objects, the object group name SHOULD be set into this attribute.

Object	Encoding	
Object Group	Text String	

Table 92: Object Group Attribute

SHALL 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

Table 93: Object Group Attribute Rules

3.29 Link

The *Link* attribute is a structure (see Table 94) used to create a link from one 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, as listed below. The *Linked Object Identifier* identifies the target Managed Cryptographic Object by its Unique Identifier. The link contains information about the association between the Managed Cryptographic Objects (e.g., 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).

Deleted: Table 94

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 contained in 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 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 that there SHALL be only one such replacement object per Managed Object.
- *Replaced Object Link.* For a Symmetric 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.

Comment: EBB: I don't think this reads right, the private key for each public key in a chain will not be available

Comment: MBJ: proposed resolution: No change.

Note that it is possible for a Managed Object to have multiple instances of the Link attribute (e.g., a Private Key has links to the associated certificate, as well as the associated public key; a Certificate object has links to both the public key and to the certificate of the certification authority (CA) that signed the certificate).

It is also possible that a Managed Object does not have links to associated cryptographic objects. This MAY occur in cases where the associated key material is not available to the server or client (e.g., the registration of a CA Signer certificate with a server, where the corresponding private key is held in a different manner).

Object	Encoding	REQUIRED
Link	Structure	
Link Type	Enumeration, see 9.1.3.2.19	Yes
Linked Object Identifier	Text String	Yes

Deleted: 9.1.3.2.19

Table 94: Link Attribute Structure

SHALL 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 Key Pair, Derive Key, Certify, Re-certify, Re-key
Applies to Object Types	All Cryptographic Objects

Table 95: Link Attribute Structure Rules

3.30 Application Specific Information

The *Application Specific Information* attribute is a structure (see Table 96) used to store data specific to the application(s) using the Managed Object. It consists of the following fields: an *Application Namespace* and *Application Data* specific to that application namespace.

Clients MAY request to set (i.e., using any of the operations that results in generating new Managed Object(s) or adding/modifying the attribute of an existing Managed Object) an instance of this attribute with a particular Application Namespace while omitting Application Data. In that case, if the server supports this namespace (as indicated by the Query operation in Section 4.24), then it SHALL return a suitable Application Data value. If the server does not support this namespace, then an error SHALL be returned.

Deleted: Table 96

Comment: EBB: I didn't see anything in the profile document

Comment: MBJ: proposed resolution: Removed the sentence. A mechanism for registration of Application Namespaces will be provided in the future.

Deleted: A list of standard application namespaces is provided in **[KMIP-Prof]**.

Deleted: 4.24

Object	Encoding	REQUIRED
Application Specific Information	Structure	
Application Namespace	Text String	Yes
Application Data	Text String	Yes

Table 96: Application Specific Information Attribute

SHALL always have a value	No
Initially set by	Client or Server (only if the Application Data is omitted, in the client request)
Modifiable by server	Yes (only if the Application Data is omitted in the client request)
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

Table 97: Application Specific Information Attribute Rules

3.31 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	
Contact Information	Text String	

Table 98: Contact Information Attribute

SHALL 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

Table 99: Contact Information Attribute Rules

3.32 Last Change Date

The *Last Change Date* attribute is a meta attribute that contains the date and time of the last change to the contents or attributes of the specified object.

Object	Encoding	
Last Change Date	Date-Time	

Table 100: Last Change Date Attribute

SHALL 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, Archive, Recover, Certify, Re-certify, Re-key, Add Attribute, Modify Attribute, Delete Attribute, Get Usage Allocation
Applies to Object Types	All Objects

Table 101: Last Change Date Attribute Rules

3.33 Custom Attribute

A *Custom Attribute* is a client- or server-defined attribute intended for vendor-specific purposes. It is created by the client and not interpreted by the server, or is created by the server and MAY be interpreted by the client. All custom attributes created by the client SHALL adhere to a naming scheme, where the name of the attribute SHALL have a prefix of 'x-'. All custom attributes created by the key management server SHALL adhere to a naming scheme where the name of the attribute SHALL have a prefix of 'y-'. The server SHALL NOT accept a client-created or modified attribute, where the name of the attribute has a prefix of ‘y-‘. The tag type Custom Attribute is not able to identify the particular attribute; hence such an attribute SHALL only appear in an Attribute Structure with its name as defined in Section 2.1.1.

Deleted: 2.1.1

Object	Encoding	
Custom Attribute	Any data type or structure	The name of the attribute SHALL start with 'x-' or 'y-'.

Table 102 Custom Attribute

SHALL 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

Table 103: Custom Attribute Rules

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 SHALL be capable of understanding the response to the request. All Object Management operations are issued in requests from clients to servers, and results obtained in responses from servers to clients. Multiple operations MAY be combined within a batch, resulting in a single request/response message pair.

Deleted: These

Deleted: into

Deleted: which allows multiple operations to be contained in

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

The key management server SHALL 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 SHALL be discarded and/or invalidated by the server, so that subsequent requests do not find this previous ID Placeholder available.

Deleted: is

The ID Placeholder is obtained from the Unique Identifier returned in response to 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 SHALL copy this Unique Identifier into the ID Placeholder variable, where it is held until the completion of the operations remaining in the batched request or until a subsequent operation in the batch causes the ID Placeholder to be replaced. If the Batch Error Continuation Option is set to Stop and the Batch Order Option is set to true, then subsequent operations in the batched request MAY make use of the ID Placeholder by omitting the Unique Identifier field from the request payloads for these operations.

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 name is specified, and there is a conflict between the single-instance attributes in the templates, then the value in the last of the conflicting templates takes precedence. If there is a conflict between the single-instance attributes in the request and the single-instance attributes in a specified template, then 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.

Deleted: 2.1.8

Deleted: subsequent

Responses MAY contain attribute values that were not specified in the request, but have been implicitly set by the server. This information is specified with a Template-Attribute that contains one or more individual attributes.

For any operations that operate on Managed Objects already stored on the server, any archived object SHALL first be made available by a Recover operation (see Section 4.22) before they MAY be specified (i.e., as on-line objects).

Comment: EBB: made available by? This assumes that an object is archived after the end of its "lifetime", rather than earlier. Is this the way all server's operate? If so, this is an assumption about server operation that needs to be stated

Deleted: moved back on-line through

Deleted: 4.22

Comment: MBJ: proposed resolution: No change. In the specification of the Archive operation, it is stated that it is up to the server if it archives objects or not. If an object is archived in the first place, it implies that archiving is supported by the server.

4.1 Create

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

Deleted: 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 that already exist.

The response contains the Unique Identifier of the created object. The server SHALL copy the Unique Identifier returned by this operation into the ID Placeholder variable.

Request Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Determines the type of object to be created.
Template-Attribute, see 2.1.8	Yes	Specifies desired object attributes using templates and/or individual attributes.

Deleted: 3.3

Deleted: 2.1.8

Table 104: Create Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Type of object created.
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the newly created object.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Deleted: 3.3

Deleted: 3.1

Deleted: 2.1.8

Table 105: Create Response Payload

Table 106 indicates which attributes SHALL be included in the Create request using the Template-Attribute object.

Deleted: Table 106

Attribute	REQUIRED
Cryptographic Algorithm, see 3.4	Yes
Cryptographic Usage Mask, see 3.14	Yes

Deleted: 3.4

Deleted: 3.14

Table 106: Create Attribute Requirements

4.2 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 MAY 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 SHALL be set to the Unique Identifier of the Private Key.

Request Payload		
Object	**REQUIRED**	**Description**
Common Template-Attribute, see 2.1.8~~2.1.8~~	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, see 2.1.8~~2.1.8~~	No	Specifies templates and/or attributes that apply to the Private Key Object. Order of precedence applies.
Public Key Template-Attribute, see 2.1.8~~2.1.8~~	No	Specifies templates and/or attributes that apply to the Public Key Object. Order of precedence applies.

Table 107: Create Key Pair Request Payload

For multi-instance 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-instance 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~~ last value of the single-instance attribute that conflict takes precedence.

Response Payload		
Object	**REQUIRED**	**Description**
Private Key Unique Identifier, see **Error! Reference source not found.**~~3.1~~	Yes	The Unique Identifier of the newly created Private Key object.
Public Key Unique Identifier, see **Error! Reference source not found.**~~3.1~~	Yes	The Unique Identifier of the newly created Public Key object.
Private Key Template-Attribute, see 2.1.8~~2.1.8~~	No	An OPTIONAL list of attributes, for the Private Key Object, with values that were not specified in the request, but have been implicitly set by the key management server.
Public Key Template-Attribute, see 2.1.8~~2.1.8~~	No	An OPTIONAL list of attributes, for the Public Key Object, with values that were not specified in the request, but have been implicitly set by the key management server.

Table 108: Create Key Pair Response Payload

Table 109~~Table 109~~ indicates which attributes SHALL be included in the Create Key pair request using Template-Attribute objects, as well as which attributes SHALL have the same value for the Private and Public Key.

Attribute	REQUIRED	SHALL contain the same value for both Private and Public Key
Cryptographic Algorithm, see 3.4	Yes	Yes
Cryptographic Length, see 3.5	Yes	Yes
Cryptographic Usage Mask, see 3.14	Yes	No
Cryptographic Domain Parameters, see 3.7	No	Yes
Cryptographic Parameters, see 3.6	No	Yes

Deleted: 3.4

Comment: EBB: The private and public key are not always the same length

Comment: MBJ: Proposed resolution: TBD

Deleted: 3.5

Deleted: 3.14

Deleted: 3.7

Deleted: 3.6

Table 109: Create Key Pair Attribute Requirements

4.3 Register

This operation requests the server to register a Managed Object that was 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 that are not to be stored by the key management system MAY be omitted from the request (e.g., private keys).

Deleted: ,

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 SHALL copy the Unique Identifier returned by this operations into the ID Placeholder variable. The Initial Date attribute of the object SHALL be set to the current time.

Request Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Determines the type of object being registered.
Template-Attribute, see 2.1.8	Yes	Specifies desired object attributes using templates and/or individual attributes.
Certificate, Symmetric Key, Private Key, Public Key, Split Key, Secret Data or Opaque Object, see 2.2	No	The object being registered. The object and attributes MAY be wrapped. Some objects (e.g., Private Keys), MAY be omitted from the request.

Deleted: 3.3

Deleted: 2.1.8

Deleted: 2.2

Table 110: Register Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the newly registered object.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Deleted: 3.1

Deleted: 2.1.8

Comment: EBB: could it also include attributes that WERE included in the request? If values are implicitly set by the server, are they required to be sent, or is that up to the server?
MBJ: same comment applies to many other response payloads in subsequent subsections

Comment: MBJ: proposed resolution: No change. As the text now reads, attributes that WERE included in the request may not be included. The TC decided to forbid mutating attributes, and to make it optional to return the list of implicitly set attributes.

Table 111: Register Response Payload

If a Managed Cryptographic Object is registered, then the following attributes SHALL be included in the Register request, either explicitly, or via specification of a template that contains the attribute.

Attribute	REQUIRED
Cryptographic Algorithm, see 3.4	Yes, MAY be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data. If present, then Cryptographic Length below SHALL also be present.
Cryptographic Length, see 3.5	Yes, MAY be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data. If present, then Cryptographic Algorithm above SHALL also be present.
Cryptographic Usage Mask, see 3.14	Yes.

Deleted: 3.4

Deleted: 3.5

Deleted: 3.14

Table 112: Register Attribute Requirements

4.4 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 attributes of the replacement key are copied from the existing key, with the exception of the attributes listed in Table 114.

Deleted: Table 114

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

Comment: EBB: Is this what is meant? Should "SHALL" replace "SHOULD" here?

Comment: MBJ: Proposed resolution: No change. It is only a recommendation, so SHOULD is correct.

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

As a result of Re-key, the Link attribute of the existing key is set to point to the replacement key and vice versa.

An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date of the replacement key. If Offset is set and dates exist for the existing key, then the dates of the replacement key SHALL be set based on the dates of the existing key as follows:

Attribute in Existing Key	Attribute in Replacement 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)

Table 113: Computing New Dates from Offset during Re-key

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

Attribute	Action
Initial Date, see 3.18	Set to the current time
Destroy Date, see 3.23	Not set
Compromise Occurrence Date, see 3.24	Not set
Compromise Date, see 3.25	Not set
Revocation Reason, see 3.26	Not set
Unique Identifier, see **Error! Reference source not found.**	New value generated
Usage Limits, see 3.16	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, see 3.2	Set to the name(s) of the existing key; all name attributes are removed from the existing key
State, see 3.17	Set based on attributes values, such as dates, as shown in Table 113
Digest, see 3.12	Recomputed from the replacement key value
Link, see 3.29	Set to point to the existing key as the replaced key
Last Change Date, see 3.32	Set to current time

Table 114: Re-key Attribute Requirements

Deleted: 3.18

Deleted: 3.23

Deleted: 3.24

Deleted: 3.25

Deleted: 3.26

Deleted: 3.1

Deleted: 3.16

Deleted: 3.2

Deleted: of

Deleted: are removed

Deleted: 3.17

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Deleted: Table 113

Deleted: 3.12

Deleted: new

Deleted: 3.29

Deleted: 3.32

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the existing Symmetric Key being re-keyed. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Offset	No	An Interval object indicating the difference between the Initialization Date and the Activation Date of the replacement key to be created.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 2.1.8

Table 115: Re-key Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the newly-created replacement Symmetric Key.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Deleted: 3.1

Deleted: 2.1.8

Table 116: Re-key Response Payload

4.5 Derive Key

This request is used to derive a symmetric key or Secret Data object from a key or secret data that is already known to the key management system. The request SHALL only apply to Managed Cryptographic Objects that have the Derive Key bit set in the Cryptographic Usage Mask attribute of the specified Managed Object (i.e., are able to be used for key derivation). If the operation is issued for an object that does not have this bit set, then the server SHALL return an error. For all derivation methods, the client SHALL specify the desired length of the derived key or Secret Data object using the Cryptographic Length attribute. If a key is created, then the client SHALL specify both its Cryptographic Length and Cryptographic Algorithm. If the specified length exceeds the output of the derivation method, then the server SHALL return an error. Clients MAY derive multiple keys and IVs by requesting the creation of a Secret Data object and specifying a Cryptographic Length that is the total length of the derived object. The length SHALL NOT exceed the length of the output returned by the chosen derivation method.

Deleted: using

Deleted: It

Deleted: secret

The fields in the request specify the Unique Identifiers of the keys or Secret Data objects to be used for derivation (e.g., some derivation methods MAY require multiple keys or Secret Data objects 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:

Deleted: secrets

Deleted: secrets

- *PBKDF2* – This method is used to derive a symmetric key from a password or pass phrase. The PBKDF2 method is published in **[PKCS#5]** and **[RFC2898]**.
- *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 **[SP800-108]**.
- *NIST800-108-F* – This method derives a key by computing the KDF in Feedback Mode as specified in **[SP800-108]**.
- *NIST800-108-DPI* – This method derives a key by computing the KDF in Double-Pipeline Iteration Mode as specified in **[SP800-108]**.
- *Extensions*

The server SHALL 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 SHALL copy the Unique Identifier returned by this operation into the ID Placeholder variable.

As a result of Derive Key, the Link attributes (i.e., Derived Key Link in the objects from which the key is derived, and the Derivation Base Object Link in the derived key) of all objects involved SHALL be set to point to the corresponding objects.

Request Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Determines the type of object to be created.
Unique Identifier, see **Error! Reference source not found.**	Yes. MAY be repeated	Determines the object or objects to be used to derive a new key. At most, two identifiers MAY be specified: one for the derivation key and another for the secret data. Note that the current value of the ID Placeholder SHALL NOT be used in place of a Unique Identifier in this operation.
Derivation Method, see 9.1.3.2.20	Yes	An Enumeration object specifying the method to be used to derive the new key.
Derivation Parameters, see below	Yes	A Structure object containing the parameters needed by the specified derivation method.
Template-Attribute, see 2.1.8	Yes	Specifies desired object attributes using templates and/or individual attributes; the length and algorithm SHALL always be specified for the creation of a symmetric key.

Table 117: Derive Key Request Payload

Deleted: 3.3

Deleted: 3.1

Deleted: here

Deleted: 9.1.3.2.20

Deleted: 2.1.8

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the newly derived key or Secret Data object.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Deleted: 3.1

Deleted: 2.1.8

Table 118: Derive Key Response Payload

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

Object	Encoding	REQUIRED
Derivation Parameters	Structure	Yes
Cryptographic Parameters, see 3.6	Structure	Yes, except for HMAC derivation keys.
Initialization Vector	Byte String	No, depends on PRF and mode of operation: empty IV is assumed if not provided.
Derivation Data	Byte String	Yes, unless the Unique Identifier of a Secret Data object is provided.

Deleted: 3.6

Table 119: Derivation Parameters Structure (Except PBKDF2)

Cryptographic Parameters identify the Pseudorandom Function (PRF) or the mode of operation of the PRF (e.g., if a key is to be derived using the HASH derivation method, then clients are REQUIRED to indicate the hash algorithm inside Cryptographic Parameters; similarly, if a key is to be derived using AES in CBC mode, then clients are REQUIRED to indicate the Block Cipher Mode). The server SHALL verify that the specified mode matches one of the instances of Cryptographic Parameters set for the corresponding key. If Cryptographic Parameters are omitted, then the server SHALL select the Cryptographic Parameters with the lowest Attribute Index for the specified key. If the corresponding key does not have any Cryptographic Parameters attribute, or if no match is found, then an error is returned.

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

Derivation Data is either the data to be encrypted, hashed, or HMACed. For the NIST SP 800-108 methods **[SP800-108]**, Derivation Data is Label||{0x00}||Context, where the all-zero byte is OPTIONAL.

Most derivation methods (e.g., ENCRYPT) require a derivation key and the derivation data to be used. The HASH derivation method requires either a derivation key or derivation data. Derivation data MAY either be explicitly provided by the client with the Derivation Data field or implicitly provided by providing the Unique Identifier of a Secret Data object. If both are provided, then an error SHALL be returned.

Deleted: encrypted

The PBKDF2 derivation method requires two additional parameters:

Object	Encoding	REQUIRED
Derivation Parameters	Structure	Yes
Cryptographic Parameters, see 3.6	Structure	No, depends on the PRF.
Initialization Vector	Byte String	No, depends on the PRF

Deleted: 3.6

		and mode of operation: an empty IV is assumed if not provided.
Derivation Data	Byte String	Yes, unless the Unique Identifier of a Secret Data object is provided.
Salt	Byte String	Yes
Iteration Count	Integer	Yes

Comment: EBB: there is no IV defined for PBKDF2

Comment: MBJ: TBD

Table 120: PBKDF2 Derivation Parameters Structure

4.6 Certify

This request is used to generate a Certificate object for a public key. This request supports certification of a new public key as well as certification of a public key that has already been certified (i.e., certificate update). Only a single certificate SHALL 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 (CA). If the server does not support this operation, an error SHALL be returned.

The Certificate Requests is passed as a Byte String, which allows multiple certificate request types for X.509 certificates (e.g., PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

Deleted: are

Deleted: s

The generated 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 generated certificate SHALL be set to point at each other.

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

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**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	The Unique Identifier of the Public Key being certified. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Certificate Request Type, see 9.1.3.2.21	Yes	An Enumeration object specifying the type of certificate request.
Certificate Request	Yes	A Byte String object with the certificate request.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 9.1.3.2.21

Deleted: 2.1.8

Table 121: Certify Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the generated Certificate object.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Deleted: 3.1

Deleted: 2.1.8

Table 122: Certify Response Payload

4.7 Re-certify

This request is used to renew an existing certificate for the same key pair. Only a single certificate SHALL be renewed at a time. Server support for this operation is OPTIONAL, as it requires that the key management system to have access to a certification authority (CA). If the server does not support this operation, an error SHALL be returned.

Deleted: with the

The Certificate Request is passed as a Byte String, which allows multiple certificate request types for X.509 certificates (e.g., PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

Deleted: s are

Deleted: s

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

If the information in the Certificate Request field in the 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 (existing) certificate.

The Link attribute of the existing certificate and of the new certificate are set to point at each other. The Link attribute of the Public Key is changed to point to the new certificate.

An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date of the new certificate. If Offset is set, then the dates of the new certificate SHALL be set based on the dates of the existing certificate (if such dates 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)

Table 123: Computing New Dates from Offset during Re-certify

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

Attribute	Action
Initial Date, see 3.18	Set to current time
Destroy Date, see 3.23	Not set
Revocation Reason, see 3.26	Not set
Unique Identifier, see 3.2	New value generated
Name, see 3.2	Set to the name(s) of the existing certificate; all name attributes are removed from the existing certificate
State, see 3.17	Set based on attributes values, such as dates, as shown in Table 123
Digest, see 3.12	Recomputed from the new certificate value.
Link, see 3.29	Set to point to the existing certificate as the replaced certificate.
Last Change Date, see 3.32	Set to current time

Deleted: 3.18

Deleted: 3.23

Deleted: 3.26

Deleted: 3.2

Deleted: 3.2

Deleted: of

Deleted: are removed

Deleted: 3.17

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Deleted: Table 123

Deleted: 3.12

Deleted: 3.29

Deleted: 3.32

Table 124: Re-certify Attribute Requirements

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	The Unique Identifier of the Certificate being renewed. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Certificate Request Type, see 9.1.3.2.21	Yes	An Enumeration object specifying the type of certificate request.
Certificate Request	Yes	A Byte 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, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

Deleted: 3.1

Formatted: Font: Not Italic

Deleted: is

Deleted: substituted by the server

Deleted: 9.1.3.2.21

Deleted: 2.1.8

Table 125: Re-certify Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the new certificate.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Deleted: 3.1

Deleted: 2.1.8

Table 126: Re-certify Response Payload

4.8 Locate

This operation requests that the server search for one or more Managed Objects, specified by one or more attributes. All attributes are allowed to be used. However, Attribute Index values SHOULD NOT be specified in the request. Attribute Index values that are provided SHALL be ignored by the Locate operation. The request MAY also contain a *Maximum Items* field, which specifies the maximum number of objects to be returned. 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.

Deleted: no attributes

Deleted: SHOULD contain Attribute Index values

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

Comment: EBB: Does this mean that the server can choose to include only one success, even if there are multiple? Or do you want all returned if there are multiple (up to some practical limit)? Or should "MAY contain" be changed to "contains"?

Comment: MBJ: proposed resolution: No change. The server can choose to return only one match even if there are multiple (server policy).

The server returns a list of Unique Identifiers of the found objects, which then MAY be retrieved using the Get operation. If the objects are archived, then the Recover and Get operations are REQUIRED to be used to obtain those objects. If a single Unique Identifier is returned to the client, then the server SHALL 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 SHALL empty the ID Placeholder, causing any subsequent operations that are batched with the Locate, and which do not specify a Unique Identifier explicitly, to fail. This ensures that these batched operations SHALL proceed only if a single object is returned by Locate.

Deleted: NOT set

Deleted: value

When using the Name or Object Group attributes for identification, wild-cards or regular expressions (defined, e.g., in **[ISO/IEC 9945-2]**) MAY be supported by specific key management system implementations.

The Date attributes in the Locate request (e.g., Initial Date, Activation Date, etc) are used to specify a time or a time range for the search. If a single instance of a given Date attribute is used in the request (e.g., the Activation Date), then objects with the same Date attribute are considered to be matching candidate objects. If two instances of the same Date attribute are used (i.e., with two different values specifying a range), then objects for which the Date attribute is inside or at a limit of the range are considered to be matching candidate objects. If a Date attribute is set to its largest possible value, then it is equivalent to an undefined attribute. The KMIP Usage Guide **[KMIP-UG]** provides examples.

When the Cryptographic Usage Mask attribute is specified in the request, candidate objects are compared against this field via an operation that consists of a logical AND of the requested mask with the mask in the candidate object, and then a comparison of the resulting value with the requested mask. For example, if the request contains a mask value of 10001100010000, and a candidate object mask contains 10000100010000, then the logical AND of the two masks is 10000100010000, which is compared against the mask value in the request (10001100010000) and the match fails. This means that a matching candidate object has all of the bits set in its mask that are set in the requested mask, but MAY have additional bits set.

Deleted: fails

Deleted: and

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

When an attribute that is defined as a structure is specified, all of the structure fields are not REQUIRED to be specified. For instance, for the Link attribute, if the Linked Object Identifier value is specified without the Link Type value, then matching candidate objects 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, only archived objects, or both on-line and archived objects are to be searched. Note that the server MAY store attributes of archived objects in order to expedite Locate operations that search through archived objects.

Deleted: 9.1.3.3.2

Request Payload		
Object	**REQUIRED**	**Description**
Maximum Items	No	An Integer object that indicates the maximum number of object identifiers the server MAY return.
Storage Status Mask, see 9.1.3.3.2	No	An Integer object (used as a bit mask) that indicates whether only on-line objects, only archived objects, or both on-line and archived objects are to be searched. If omitted, then on-line only is assumed.
Attribute, see 3	Yes, MAY be repeated	Specifies an attribute and its value(s) that are REQUIRED to match those in a candidate object (according to the matching rules defined above).

Deleted: SHALL

Deleted: 9.1.3.3.2

Deleted: 3

Deleted:

Deleted: are REQUIRED to match the desired object.

Table 127: Locate Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No, MAY be repeated	The Unique Identifier of the located objects.

Deleted: 3.1

Table 128: Locate Response Payload

4.9 Check

This operation requests that the server check for the use of a Managed Object according to 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.

Deleted: 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 according to the specified attributes, then the server returns the Unique Identifier of the object.

If the server determines that the client is not allowed to use the object according to the specified attributes, then the server empties the ID Placeholder and does not return the Unique Identifier, and the operation returns the set of attributes specified in the request that caused the server policy denial. The only attributes returned are those that resulted in the server determining that the client is not allowed to use the object, thus allowing the client to determine how to proceed. The operation also returns a failure, and the server SHALL ignore any subsequent operations in the batch.

Deleted: invalidates

Deleted: value

The additional objects that MAY be specified in the request are limited to:

- Usage Limits Byte Count or Usage Limits Object Count (see Section 3.16) – 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.

Deleted: 3.16

- Cryptographic Usage Mask – This is used to specify the cryptographic operations for which the client intends to use the object (see Section 3.14). 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 MAY be used for both Encryption and Decryption, but the value in the Check operation MAY specify that the client is only using the key for Encryption at this time.

Deleted: 3.14

- Lease Time – This specifies a desired lease time (see Section 3.15). The client MAY use this to determine if the server allows 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 MAY be granted if requested by a subsequent, batched, Obtain Lease operation.

Deleted: 3.15

Note that these objects are not encoded in an Attribute structure as shown in Section 2.1.1.

Deleted: 2.1.1

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object being checked. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Usage Limits Byte Count, see 3.16	No	Specifies the number of bytes to be protected to be checked against server policy. SHALL NOT be present if the Usage Limits Object Count is present.
Usage Limits Object Count, see 3.16	No	Specifies the number of objects to be protected to be checked against server policy. SHALL NOT be present if the Usage Limits Byte Count is present.
Cryptographic Usage Mask, see 3.14	No	Specifies the Cryptographic Usage for which the client intends to use the object.
Lease Time, see 3.15	No	Specifies a Lease Time value that the Client is asking the server to validate against server policy.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 3.16

Deleted: 3.16

Deleted: 3.14

Deleted: 3.15

Table 129: Check Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object.
Usage Limits Byte Count, see 3.16	No	Returned by the Server if the Usage Limits value specified in the Request Payload is larger than the value that the server policy allows. SHALL NOT be present if the Usage Limits Object Count is present.
Usage Limits Object Count, see 3.16	No	Returned by the Server if the Usage Limits value specified in the Request Payload is larger than the value that the server policy allows. SHALL NOT

Deleted: 3.1

Deleted: 3.16

Deleted: 3.16

		be present if the Usage Limits Byte Count is present.
Cryptographic Usage Mask, see 3.14	No	Returned by the Server if the Cryptographic Usage Mask specified in the Request Payload is rejected by the server for policy violation.
Lease Time, see 3.15	No	Returned by the Server if the Lease Time value in the Request Payload is larger than a valid Lease Time that the server MAY grant.

Deleted: 3.14

Deleted: 3.15

Table 130: Check Response Payload

The encodings of the Usage limits Byte and Object Counts is as shown in Section 3.16

Deleted: 3.16

4.10 Get

This operation requests that the server returns the Managed Object specified by its Unique Identifier.

Deleted: in the request

Deleted: 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. The response contains the Unique Identifier of the object, along with the object itself, which MAY be wrapped using a wrapping key as specified in the request.

The following key format capabilities SHALL be assumed by the client restrictions apply when the client requests the server to return an object in a particular format:

Deleted: requesting

- If a client registered a key in a given format, the server SHALL be able to return the key during the Get operation in the same format that was used when the key was registered.
- Any other format conversion MAY optionally be supported by the server.

Deleted: registers

Deleted: at least that

Deleted: as it

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object being requested. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Key Format Type, see 9.1.3.2.3	No	Determines the key format type to be returned
Key Compression Type, see 9.1.3.2.2	No	Determines the compression method for elliptic curve public keys
Key Wrapping Specification, see 2.1.6	No	Specifies keys and other information for wrapping the returned object. This field SHALL NOT be specified if the requested object is a Template.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 9.1.3.2.3

Deleted: 9.1.3.2.2

Deleted: 2.1.6

Table 131: Get Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Type of object
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object
Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template, Secret Data, or Opaque Object, see 2.2	Yes	The cryptographic object being returned

Deleted: 3.3

Deleted: 3.1

Deleted: 2.2

Table 132: Get Response Payload

4.11 Get Attributes

This operation requests one or more attributes of a Managed Object. The object is specified by its Unique Identifier and the attributes are specified by their name in the request. If a specified attribute has multiple instances, then all instances are returned. If a specified attribute does not exist (i.e., has no value), then it SHALL NOT be present in the returned response. If no requested attributes exist, then the response SHALL consist only of the Unique Identifier.

Deleted: returns

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object whose attributes are being requested. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier
Attribute Name, see 2.1.1	Yes, MAY be repeated	Specifies a desired attribute of the object

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 2.1.1

Table 133: Get Attributes Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object
Attribute, see 2.1.1	No, MAY be repeated	The requested attribute for the object

Deleted: 3.1

Deleted: 2.1.1

Table 134: Get Attributes Response Payload

4.12 Get Attribute List

This operation requests a list of the attribute names associated with a Managed Object. The object is specified by its Unique Identifier.

Deleted: returns

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object whose attribute names are being requested. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier

Deleted: 3.1

Deleted: is substituted by the server

Table 135: Get Attribute List Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see Error! Reference source not found.	Yes	The Unique Identifier of the object
Attribute Name, see 2.1.1	Yes, MAY be repeated	The names of the available attributes for the object

Table 136: Get Attribute List Response Payload

Deleted: 3.1

Deleted: 2.1.1

Deleted: requested

Deleted: names

4.13 Add Attribute

This request adds a new attribute instance to a Managed Object and sets its value. The request contains the Unique Identifier of the Managed Object to which the attribute pertains, along with the attribute name and value. For non-multi-instance attributes, this is how the attribute value is created. For multi-instance attributes, this is how the first and subsequent values are created. Existing attribute values SHALL only be changed by the Modify Attribute operation. Read-Only attributes SHALL NOT be added using the Add Attribute operation. No Attribute Index SHALL be specified in the request. The response returns a new Attribute Index, although the Attribute Index MAY be omitted if the index of the added attribute instance is 0. Multiple Add Attribute requests MAY be included in a single batched request to add multiple attributes.

Deleted: and

Deleted:

Deleted: y

Deleted: are

Deleted: if the attribute being added is allowed to have multiple instances

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see Error! Reference source not found.	No	The Unique Identifier of the object. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Attribute, see 2.1.1	Yes	Specifies the attribute to be added for the object.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 2.1.1

Deleted: of the object

Table 137: Add Attribute Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see Error! Reference source not found.	Yes	The Unique Identifier of the object
Attribute, see 2.1.1	Yes	The added attribute

Deleted: 3.1

Deleted: 2.1.1

Table 138: Add Attribute Response Payload

4.14 Modify Attribute

This request modifies the value of an existing attribute instance associated with a Managed Object. The request contains the Unique Identifier of the Managed Object whose attribute is to be modified, and the attribute name, OPTIONAL Attribute Index, and the new value. Only existing attributes MAY be changed via this operation. New attributes SHALL only be added by the Add Attribute operation. If an Attribute Index is specified, then only the specified instance of the attribute is modified. If the attribute has multiple instances, and no Attribute Index is specified in the request, then the Attribute Index is assumed to be 0. If the attribute does not support multiple instances, then the Attribute Index SHALL NOT be specified. Specifying an Attribute Index for which there exists no Attribute Value SHALL result in an error.

Deleted: Read-Only attributes SHALL NOT be changed using this operation.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	The Unique Identifier of the object. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Attribute, see 2.1.1	Yes	Specifies the attribute of the object to be modified.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 2.1.1

Table 139: Modify Attribute Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object
Attribute, see 2.1.1	Yes	The modified attribute

Deleted: 3.1

Deleted: 2.1.1

Table 140: Modify Attribute Response Payload

4.15 Delete Attribute

This request deletes an attribute associated with a Managed Object. The request contains the Unique Identifier of the Managed Object whose attribute is to be deleted, the attribute name, and optionally the Attribute Index of the attribute. Attributes that SHALL always have a value SHALL never be deleted by this operation. If no Attribute Index is specified, and the Attribute whose name is specified has multiple instances, then the operation is rejected. Note that only a single attribute instance SHALL be deleted at a time. Multiple delete operations (e.g., possibly batched) are necessary to delete several attribute instances. Attempting to delete a non-existent attribute or specifying an Attribute Index for which there exists no Attribute Value SHALL result in an error.

Deleted: REQUIRED attributes and Read-Only attributes SHALL NOT be deleted by this operation.

Deleted: s

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object whose attributes are being deleted. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Attribute Name, see 2.1.1	Yes	Specifies the name of the attribute to be deleted.
Attribute Index, see 2.1.1	No	Specifies the Index of the Attribute.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 2.1.1

Deleted: 2.1.1

Table 141: Delete Attribute Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object
Attribute, see 2.1.1	Yes	The deleted attribute

Deleted: 3.1

Deleted: 2.1.1

Table 142: Delete Attribute Response Payload

4.16 Obtain Lease

Comment: EBB suggestion to rename operation to "Obtain Lease Time"

Comment: MBJ: proposed resolution: No change.

This request is used to obtain a new *Lease Time* for a specified Managed Object. The Lease Time is an interval value that determines when the client's internal cache of information about the object expires and needs to 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, then the client SHALL NOT use the associated cryptographic object until a new lease is obtained. If the server determines that a new lease SHALL NOT be issued for the specified cryptographic object, then the server SHALL respond to the Obtain Lease request with an error.

The response payload for the operation contains the current value of the Last Change 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.

Deleted: also

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object for which the lease is being obtained. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Deleted: 3.1

Formatted: Font: Not Italic

Deleted: is substituted by the server

Table 143: Obtain Lease Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object.
Lease Time, see 3.15	Yes	An interval (in seconds) that specifies the amount of time that the object MAY be used until a new lease needs to be obtained.
Last Change Date, see 3.32	Yes	The date and time indicating when the latest change was made to the contents or any attribute of the specified object.

Deleted: 3.1

Deleted: 3.15

Deleted: 3.32

Table 144: Obtain Lease Response Payload

4.17 Get Usage Allocation

Comment: EBB: Explain the principal and reason for this here.

Comment: MBJ: proposed resolution: No change. Possibly UG item.

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 applying cryptographic protection. The allocation only applies to Managed Cryptographic Objects that are able to be used for applying protection (e.g., symmetric keys for encryption, private keys for signing, etc.) and is only valid if the Managed Cryptographic Object has a Usage Limits attribute. Usage for processing cryptographically-protected information (e.g., decryption, verification, etc.) is not limited and is not able to be allocated. A Managed Cryptographic Object that has a Usage Limits attribute SHALL NOT be used by a client for applying cryptographic protection unless an allocation has been obtained using this operation. The operation SHALL only be requested 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 requested for an object that has no Usage Limits attribute, or is not an object that MAY be used for applying cryptographic protection, then the server SHALL return an error.

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 SHALL be specified in the request. If the requested amount is not available or if the Managed Object is not able to be used for applying cryptographic protection at this time, then the server SHALL return an error. The server SHALL assume that the entire allocated amount will be

Deleted: has been

consumed. Once the entire allocated amount has been consumed, the client SHALL NOT continue to use the Managed Cryptographic Object for applying cryptographic protection until a new allocation is obtained.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object whose usage allocation is being requested. If omitted, then the ID Placeholder is substituted by the server.
Usage Limits Byte Count, see 3.16	No	The number of bytes to be protected. SHALL be present if the Usage Limits Object Count is not present.
Usage Limits Object Count, see 3.16	No	The number of objects to be protected. SHALL be present if the Usage Limits Byte Count is not present.

Deleted: 3.1

Deleted: 3.16

Deleted: 3.16

Table 145: Get Usage Allocation Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object.

Deleted: 3.1

Table 146: Get Usage Allocation Response Payload

4.18 Activate

This request is used to activate a Managed Cryptographic Object. The request SHALL NOT specify a Template object. The operation SHALL only be performed on an object in the Pre-Active state and has the effect of changing its state to Active, and setting its Activation Date to the current date and time.

Deleted: The request contains the Unique Identifier of the Managed Cryptographic Object.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object being activated. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier

Deleted: 3.1

Deleted: is substituted by the server

Table 147: Activate Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object

Deleted: 3.1

Table 148: Activate Response Payload

4.19 Revoke

This request is used to revoke a Managed Cryptographic Object or an Opaque Object. The request SHALL NOT specify a Template object. The request contains a reason for the revocation (e.g., “compromised”, “no longer used”, etc). Special authentication and authorization SHOULD be enforced to perform this request (see **[KMIP-UG]**). Only the object creator or an authorized security officer SHOULD

Deleted: the unique identifier of the Managed Cryptographic Object and be allowed to issue this request. The operation has one of two effects. If the revocation reason is “compromised”, then the object is placed into the “compromised” state, and the Compromise Date attribute is set to the current date and time. Otherwise, the object is placed into the “deactivated” state, and the Deactivation Date attribute is set to the current date and time.

Comment: EBB: This sentence needs too be turned around (according to Matt Ball): Entities other than the object creator or an authorized security officer SHOULD NOT be allowed to issue this request.

Comment: MBJ: proposed resolution: No change.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object being revoked. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Revocation Reason, see 3.26	Yes	Specifies the reason for revocation.
Compromise Occurrence Date, see 3.24	No	SHALL be specified if the Revocation Reason is 'compromised'.

Deleted: 3.1

Deleted: is substituted by the server

Deleted: 3.26

Deleted: 3.24

Table 149: Revoke Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object

Deleted: 3.1

Table 150: Revoke Response Payload

4.20 Destroy

This request is used to indicate to the server that the key material for the specified Managed Object SHALL be destroyed. The meta-data for the key material MAY be retained by the server (e.g., used to ensure that an expired or revoked private signing key is no longer available). Special authentication and authorization SHOULD be enforced to perform this request (see **[KMIP-UG]**). Only the object creator or an authorized security officer SHOULD be allowed to issue this request. If the Unique Identifier specifies a Template object, then the object itself, including all meta-data, SHALL be destroyed.

Comment: EBB: Conflicts with the last sentence of this paragraph

Comment: MBJ: proposed resolution: No change. Template objects have no key material, so the last sentence does not apply to this sentence.

Comment: EBB: This sentence needs too be turned around (according to Matt Ball): Entities other than the object creator or an authorized security officer SHOULD NOT be allowed to issue this request.

Comment: MBJ: proposed resolution: No change.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object being destroyed. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Deleted: 3.1

Deleted: is substituted by the server

Table 151: Destroy Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object

Deleted: 3.1

Table 152: Destroy Response Payload

4.21 Archive

This request is used to specify that a Managed Object MAY be archived. The actual time when the object is archived, 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 Managed Object. Special authentication and authorization SHOULD be enforced to perform this request (see **[KMIP-UG]**). Only the object creator or an authorized security officer SHOULD be allowed to issue this request. This request is only an indication from a client that from its point of view it is possible for the key management system to archive the object.

Comment: EBB: see previous comment

Comment: MBJ: proposed resolution: No change.

Deleted: a "hint" to

Deleted: possibly

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object being archived. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Deleted: 3.1

Deleted: is substituted by the server

Table 153: Archive Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object

Deleted: 3.1

Table 154: Archive Response Payload

4.22 Recover

This request is used to obtain access to a Managed Object that has been archived. This request MAY require asynchronous polling to obtain the response due to delays caused by retrieving the object from the archive. Once the response is received, the object is now on-line, and MAY be obtained (e.g., via a Get operation). Special authentication and authorization SHOULD be enforced to perform this request (see **[KMIP-UG]**).

Comment: EBB: how long can it remain online? I would expect that there are lots of cases where this should be taken offline relatively quickly

Comment: MBJ: proposed resolution: No change.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	No	Determines the object being recovered. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Deleted: 3.1

Deleted: is substituted by the server

Table 155: Recover Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object

Deleted: 3.1

Table 156: Recover Response Payload

4.23 Validate

This requests that the server validate a certificate chain and return information on its validity. Only a single certificate chain SHALL be included in each request. Support for this operation at the server is OPTIONAL. If the server does not support this operation, an error SHALL be returned.

The request may contain a list of certificate objects, and/or a list of Unique Identifiers that identify Managed Certificate objects. Together, the two lists compose a certificate chain to be validated. The request MAY also contain a date for which all certificates in the certificate chain are REQUIRED to be valid.

Deleted: is

The method or policy by which validation is 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.

Request Payload		
Object	**REQUIRED**	**Description**
Certificate, see 2.2.1	No, MAY be repeated	One or more Certificates.
Unique Identifier, see **Error! Reference source not found.**	No, MAY be repeated	One or more Unique Identifiers of Certificate Objects.
Validity Date	No	A Date-Time object indicating when the certificate chain needs to be valid.

Deleted: 2.2.1

Deleted: 3.1

Deleted: is

Table 157: Validate Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Validity Indicator, see 9.1.3.2.22	Yes	An Enumeration object indicating whether the certificate chain is valid, invalid, or unknown.

Deleted: 9.1.3.2.22

Table 158: Validate Response Payload

4.24 Query

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

- Query Operations
- Query Objects
- Query Server Information
- Query Application Namespaces

The *Operation* fields in the response contain Operation enumerated values, which SHALL list all the operations that the server supports. If the request contains a Query Operations value in the Query Function field, then these fields SHALL be returned in the response.

The *Object Type* fields in the response contain Object Type enumerated values, which SHALL list all the object types that the server supports. If the request contains a *Query Objects* value in the Query Function field, then these fields SHALL be returned in the response.

The *Server Information* field in the response is a structure containing vendor-specific fields and/or substructures. If the request contains a *Query Server Information* value in the Query Function field, then this field SHALL be returned in the response.

The Application Namespace fields in the response contain the namespaces that the server SHALL generate values for if requested by the client (see Section 3.30). These fields SHALL only be returned in the response if the request contains a Query Application Namespaces value in the Query Function field.

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

Formatted: Bullets and Numbering

Deleted: OPTIONAL

Deleted: The OPTIONAL operations are:¶
<#>Validate¶
<#>Certify¶
<#>Re-Certify¶
<#>Notify¶
Put

Deleted: The object types (any of which are OPTIONAL) are:

Deleted: <#>Certificate¶
<#>Symmetric Key¶
<#>Public Key¶
<#>Private Key¶
<#>Split Key¶
<#>Template¶
<#>Secret Data¶
<#>Opaque Object¶

Deleted: 3.30

Request Payload		
Object	**REQUIRED**	**Description**
Query Function, see 9.1.3.2.23	Yes, MAY be Repeated	Determines the information being queried

Deleted: 9.1.3.2.23

Table 159: Query Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Operation, see 9.1.3.2.26	No, MAY be repeated	Specifies an Operation that is supported by the server. Only OPTIONAL operations SHALL be listed.
Object Type, see 3.3	No, MAY be repeated	Specifies a Managed Object Type that is supported by the server.
Vendor Identification	No	SHALL be returned if Query Server Information is requested. The Vendor Identification SHALL be a text string that uniquely identifies the vendor.
Server Information	No	Contains vendor-specific information possibly be of interest to the client.
Application Namespace, see 3.30	No, MAY be repeated	Specifies an Application Namespace supported by the server.

Deleted: 9.1.3.2.26

Deleted: 3.3

Deleted: 3.30

Table 160: Query Response Payload

4.25 Cancel

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

Deleted: 6.8

- *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 is not able to be asynchronous.

Request Payload		
Object	**REQUIRED**	**Description**
Asynchronous Correlation Value, see 6.8	Yes	Specifies the request being canceled

Deleted: 6.8

Table 161: Cancel Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Asynchronous Correlation Value, see 6.8	Yes	Specified in the request
Cancellation Result, see 9.1.3.2.24	Yes	Enumeration indicating the result of the cancellation

Deleted: 6.8

Deleted: 9.1.3.2.24

Table 162: Cancel Response Payload

4.26 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 SHALL be specified in the request. The response to this operation SHALL NOT be asynchronous.

Deleted: 6.8

Request Payload		
Object	**REQUIRED**	**Description**
Asynchronous Correlation Value, see 6.8	Yes	Specifies the request being polled

Deleted: 6.8

Table 163: Poll Request Payload

The server SHALL reply with one of two responses:

If the operation has not completed, the response SHALL contain no payload and a Result Status of Pending.

If the operation has completed, the response SHALL contain the appropriate payload for the operation. This response SHALL 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 via means outside of the normal client-server request-response mechanism. These operations are used to send Managed Cryptographic Objects directly to clients without a specific request from the client.

5.1 Notify

This operation is used to notify a client of events that resulted in changes to attributes of an object. This operation is only ever sent by a server to a client via means outside of 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 uses the same format as a Request message (see 7.1, Table 182), except that the Maximum Response Size, Asynchronous Indicator, Batch Error Continuation Option, and Batch Order Option fields are not allowed. The client SHALL send a response in the form of a Response Message (see 7.1, Table 183) containing no payload, unless both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the client is not able to respond.

Deleted: is sent as a normal

Comment: EBB: I agree that this doesn't belong in this message, but is there a possibility that the notification would be too long for the client?

Comment: MBJ: proposed resolution: Added possibility for a client to return an error (defined in Section 11) if it is able to respond

Deleted: Table 182

Deleted: Table 183

Deleted: Server and Client support for this message is OPTIONAL.

Message Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object.
Attribute, see 3	Yes, MAY be repeated	The attributes that have changed. This includes at least the Last Change Date attribute.

Deleted: 3.1

Deleted: 3

Table 164: Notify Message Payload

5.2 Put

This operation is used to "push" Managed Cryptographic Objects to clients. This operation is only ever sent by a server to a client via means outside of 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 that is being sent, and the object itself. The message uses the same format as a Request message (see 7.1, Table 182), except that the Maximum Response Size, Asynchronous Indicator, Batch Error Continuation Option, and Batch Order Option fields are not allowed. The client SHALL send a response in the form of a Response Message (see 7.1, Table 183) containing no payload, unless both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the client is not able to respond.

Deleted: is sent as

Deleted: normal

Deleted: Table 182

Deleted: Table 183

Deleted: Server and client support for this message is OPTIONAL.

The *Put Function* field indicates whether the object being "pushed" is a new object, or is a replacement for an object already known to the client (e.g., 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 SHALL contain one of the following values:

- *New* – which indicates 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 is sending along with the object. The server MAY include attributes with the object to specify how the object is to be used by the client. The server MAY include a Lease Time attribute that grants a lease to the client.

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

Message Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see **Error! Reference source not found.**	Yes	The Unique Identifier of the object.
Put Function, see 9.1.3.2.25	Yes	Indicates function for Put message.
Replaced Unique Identifier, see **Error! Reference source not found.**	No	Unique Identifier of the replaced object. SHALL be present if the *Put Function* is *Replace.*
Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template, Secret Data, or Opaque Object, see 2.2	Yes	The object being sent to the client.
Attribute, see 3	No, MAY be repeated	The additional attributes that the server wishes to send with the object.

Deleted: 3.1

Deleted: 9.1.3.2.25

Deleted: 3.1

Deleted: 2.2

Deleted: 3

Table 165: Put Message Payload

6 Message Contents

The messages in the protocol consist of a message header, 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 to which is being replied.

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

6.1 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 SHALL support backward compatibility with versions of the protocol with the same major version. Support for backward compatibility with different major versions is OPTIONAL.

Comment: EBB: Would a minor change cause compatibility problems with an implementation that implements the same major version, but an earlier minor version?

Comment: MBJ: proposed resolution: No change. The current wording prohibits this

Object	Encoding	REQUIRED
Protocol Version	Structure	
Protocol Version Major	Integer	Yes
Protocol Version Minor	Integer	Yes

Table 166: Protocol Version Structure in Message Header

6.2 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.

Deleted: 4

Deleted: 5

Object	Encoding	
Operation	Enumeration, see 9.1.3.2.26	

Deleted: 9.1.3.2.26

Table 167: Operation in Batch Item

6.3 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 SHALL handle. It SHOULD only be sent in requests that possibly return large replies.

Object	Encoding	
Maximum Response Size	Integer	

Table 168: Maximum Response Size in Message Request Header

6.4 Unique Batch Item ID

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

Object	Encoding	
Unique Batch Item ID	Byte String	

Table 169: Unique Batch Item ID in Batch Item

6.5 Time Stamp

This field is optionally contained in a client request. It is REQUIRED in a server request and response. It 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 server's time). Note that the time stamp MAY be used by a client that has no real-time clock, but has a countdown timer, to obtain useful “seconds from now” values from all of the Date attributes by performing a subtraction.

Deleted: ,

Deleted: ,

Deleted: known correct

Object	Encoding	
Time Stamp	Date-Time	

Table 170: Time Stamp in Message Header

6.6 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 requests, or all requests, depending on policy. Query operations used to interrogate server features and functions SHOULD NOT require authentication.

The authentication mechanisms are described and discussed in Section 8.

Deleted: 8

Object	Encoding	REQUIRED
Authentication	Structure	
Credential	Structure, see 2.1.2	Yes

Deleted: 2.1.2

Table 171: Authentication Structure in Message Header

6.7 Asynchronous Indicator

This Boolean flag indicates whether the client is able to accept an asynchronous response. It SHALL have the Boolean value True if the client is able to handle asynchronous responses, and the value False otherwise. If not present in a request, then False is assumed. If a client indicates that it is not able to handle asynchronous responses (i.e., flag is set to False), and the server is not able to process the request synchronously, then the server SHALL respond to the request with a failure.

Object	Encoding	
Asynchronous Indicator	Boolean	

Table 172: Asynchronous Indicator in Message Request Header

6.8 Asynchronous Correlation Value

This is returned in the immediate response to an operation that is pending and that requires asynchronous polling. Note: the server decides which operations are performed synchronously or asynchronously. A server-generated correlation value SHALL be specified in any subsequent Poll or Cancel operations that pertain to the original operation.

Comment: EBB: How does a client find out which the server can handle asynchronously?

Comment: MBJ: proposed resolution: No change. The client cannot find out.

Comment: EBB: Is there an explanation anywhere about how the asynchronous process works?

Comment: MBJ: proposed resolution: No change. Maybe add to UG

Object	Encoding	
Asynchronous Correlation Value	Byte String	

Table 173: Asynchronous Correlation Value in Response Batch Item

6.9 Result Status

This is sent in a response message and indicates the success or failure of a request. The following values MAY be set in this field:

- *Success* – The requested operation completed successfully.
- *Operation Pending* – The requested operation is in progress, and it is necessary to obtain the actual result via asynchronous polling. The asynchronous correlation value SHALL be used for the subsequent polling of the result status.
- *Operation Undone* – The requested operation was performed, but had to be undone (i.e., due to a failure in a batch for which the Error Continuation Option (see 6.13, 7.2 and 7.3) was set to Undo).
- *Operation Failed* – The requested operation failed.

Deleted: *ure*

Object	Encoding	
Result Status	Enumeration, see 9.1.3.2.27	

Deleted: 9.1.3.2.27

Table 174: Result Status in Response Batch Item

6.10 Result Reason

This field indicates a reason for failure or a modifier for a partially successful operation and SHALL be present in responses that return a Result Status of Failure. In such a case the Result Reason SHALL be set as specified in Section 11. It is OPTIONAL in any response that returns a Result Status of Success. The following defined values are defined for this field:

Comment: EBB: What reasons would be returned for successes? where are these listed, or are they vendor/implementation-specific?

Comment: MBJ: proposed resolution: No change. These would be vendor extensions.

Deleted: 11

- *Item not found* – A requested object was not found or did not exist.
- *Response too large* – The response to a request would exceed the *Maximum Response Size* in the request.
- *Authentication not successful* – The authentication information in the request was not able to be validated, or there was no authentication information 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 is not supported by the server.
- *Missing data* – The operation requires additional OPTIONAL information in the request, which was not present.
- *Invalid field* – Some data item in the request has an invalid value.
- *Feature not supported* – An OPTIONAL feature specified in the request is 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 was not able to be performed with the specified parameters.
- *Permission denied* – The client does not have permission to perform the requested operation.
- *Object archived* – The object SHALL be recovered from the archive before performing the operation.
- *Index Out of Bounds* – The client tried to set more instances than the server supports of an attribute that MAY have multiple instances.

Formatted: Bullets and Numbering

- *Application Namespace Not Supported* – The particular Application Namespace is not supported, and server was not able to generate the Application Data field of an Application Specific Information attribute if the field was omitted from the client request.
- *Key Format Type and/or Key Compression Type Not Supported* – The object exists but the server is unable to provide it in the desired Key Format Type and/or Key Compression Type.
- *General failure* – The request failed for a reason other than the defined reasons above.

Object	Encoding	
Result Reason	Enumeration, see 9.1.3.2.28	

Table 175: Result Reason in Response Batch Item

Comment: EBB: Alphabetize?

Comment: MBJ: proposed resolution: No change. Same order as the enumeration in Section 9.

Deleted: 9.1.3.2.28

6.11 Result Message

This field MAY be returned in a response. It contains a more descriptive error message, which MAY be provided to an end user or used for logging/auditing purposes.

Deleted: used by the client

Deleted: display to

Object	Encoding	
Result Message	Text String	

Table 176: Result Message in Response Batch Item

6.12 Batch Order Option

A Boolean value used in requests where the Batch Count is greater than 1. If True, then batched operations SHALL be executed in the order in which they appear within the request. If False, then the server MAY choose to execute the batched operations in any order. If not specified, then 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 batch order option set to True, then the entire request SHALL be rejected.

Object	Encoding	
Batch Order Option	Boolean	

Table 177: Batch Order Option in Message Request Header

6.13 Batch Error Continuation Option

This option SHALL only be present if the Batch Count is greater than 1. This option SHALL have one of three values:

- *Undo* – If any operation in the request fails, then the server SHALL undo all the previous operations.
- *Stop* – If an operation fails, then the server SHALL NOT continue processing subsequent operations in the request. Completed operations SHALL NOT be undone.
- *Continue* – Return an error for the failed operation, and continue processing subsequent operations in the request.

If not specified, then 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 with a value other than the default Stop, then the entire request SHALL be rejected.

Object	Encoding	
Batch Error Continuation	Enumeration, see 9.1.3.2.29	

Option		

Table 178: Batch Error Continuation Option in Message Request Header

6.14 Batch Count

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

Object	Encoding	
Batch Count	Integer	

Table 179: Batch Count in Message Header

6.15 Batch Item

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

Deleted: 7.2

Deleted: 7.3

Object	Encoding	
Batch Item	Structure	

Table 180: Batch Item in Message

6.16 Message Extension

The *Message Extension* is an OPTIONAL structure that 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, a Criticality Indicator, and vendor-specific extensions. The *Vendor Identification* SHALL be a text string that uniquely identifies the vendor, allowing a client to determine if it is able to parse and understand the extension. If a client or server receives a protocol message containing a message extension that it does not understand, then its actions depend on the *Criticality Indicator*. If the indicator is True (i.e., Critical), and the receiver does not understand the extension, then the receiver SHALL reject the entire message. If the indicator is False (i.e., Non-Critical), and the receiver does not understand the extension, then the receiver MAY process the rest of the message as if the extension were not present.

Deleted:

Object	Encoding	REQUIRED
Message Extension	Structure	
Vendor Identification	Text String	Yes
Criticality Indicator	Boolean	Yes
Vendor Extension	Structure	Yes

Table 181: Message Extension Structure in Batch Item

7 Message Format

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

7.1 Message Structure

Object	Encoding	REQUIRED
Request Message	Structure	
Request Header	Structure, see Table 184 and Table 188	Yes
Batch Item	Structure, see Table 185 and Table 189	Yes, MAY be repeated

Table 182: Request Message Structure

Object	Encoding	REQUIRED
Response Message	Structure	
Response Header	Structure, see Table 186 and Table 190	Yes
Batch Item	Structure, see Table 187 and Table 191	Yes, MAY be repeated

Table 183: Response Message Structure

7.2 Synchronous Operations

Synchronous Request Header		
Object	**REQUIRED in Message**	**Comment**
Request Header	Yes	Structure
Protocol Version	Yes	See 6.1
Maximum Response Size	No	See 6.3
Authentication	No	See 6.6
Batch Error Continuation Option	No	If omitted, then Stop is assumed, see 6.13
Batch Order Option	No	If omitted, then False is assumed, see 6.12
Time Stamp	No	See 6.5
Batch Count	Yes	See 6.14

Table 184: Synchronous Request Header Structure

Synchronous Request Batch Item

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Check spelling and grammar

Deleted: Table 184

Formatted: Font: 10 pt

Deleted: Table 188

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Check spelling and grammar

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Deleted: Table 189

Formatted: Font: 10 pt, Check spelling and grammar

Deleted: Table 185

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Check spelling and grammar

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Check spelling and grammar

Deleted: Table 186

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Check spelling and grammar

Deleted: Table 190

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted: Font: 10 pt, Check spelling and grammar

Deleted: Table 187

Deleted: Table 191

Formatted: Font: 10 pt

Formatted: Font: 10 pt

Formatted ... [1]

Deleted: 6.1

Deleted: 6.3

Deleted: 6.6

Deleted: 6.13

Deleted: 6.12

Deleted: 6.5

Deleted: 6.14

Object	REQUIRED in Message	Comment
Batch Item	Yes	Structure, see 6.15
Operation	Yes	See 6.2
Unique Batch Item ID	No	REQUIRED if *Batch Count* > 1, see 6.4
Request Payload	Yes	Structure, contents depend on the Operation, see 4and 5
Message Extension	No	See 6.16

Table 185: Synchronous Request Batch Item Structure

Synchronous Response Header		
Object	**REQUIRED in Message**	**Comment**
Response Header	Yes	Structure
Protocol Version	Yes	See 6.1
Time Stamp	Yes	See 6.5
Batch Count	Yes	See 6.14

Table 186: Synchronous Response Header Structure

Synchronous Response Batch Item		
Object	**REQUIRED in Message**	**Comment**
Batch Item	Yes	Structure, see 6.15
Operation	Yes, if not a failure	See 6.2
Unique Batch Item ID	No	REQUIRED if present in Request Batch Item, see 6.4
Result Status	Yes	See 6.9
Result Reason	Yes, if Result Status is *Failure*	Only required if Result Status is *Failure*, otherwise optional. See 6.10
Result Message	No	OPTIONAL if Result Status is not *Success*, see 6.11
Response Payload	Yes, if not a failure	Structure, contents depend on the Operation, see 4and 5
Message Extension	No	See 6.16

Table 187: Synchronous Response Batch Item Structure

7.3 Asynchronous Operations

If the client is capable of accepting asynchronous responses, then 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.

Deleted: 6.15

Deleted: 6.2

Deleted: 6.4

Deleted: 4

Deleted: 5

Deleted: 6.16

Deleted: 6.1

Deleted: 6.5

Deleted: 6.14

Deleted: 6.15

Deleted: 6.2

Deleted: 6.4

Deleted: 6.9

Deleted: present

Formatted: Font: Italic

Deleted: No

Deleted: not *Success*

Formatted: Font: Not Italic

Comment: EBB: Does not agree with 6.10; see my comment there.

Comment: MBJ: proposed resolution: Rephrased to be consistent with 6.10

Deleted: ,

Deleted: s

Deleted: 6.10

Deleted: Only present

Comment: EBB: This is not stated in either 6.10 or 6.11.

Comment: MBJ: proposed resolution: No change.

Deleted: 6.11

Deleted: 4

Deleted: 5

Deleted: 6.16

Asynchronous Request Header		
Object	**REQUIRED in Message**	**Comment**
Request Header	Yes	Structure
Protocol Version	Yes	See 6.1
Maximum Response Size	No	See 6.3
Asynchronous Indicator	Yes	SHALL be set to True, see 6.7
Authentication	No	See 6.6
Batch Error Continuation Option	No	If omitted, then Stop is assumed, see 6.13
Batch Order Option	No	If omitted, then False is assumed, see 6.12
Time Stamp	No	See 6.5
Batch Count	Yes	See 6.14

Deleted: 6.1

Deleted: 6.3

Deleted: 6.7

Deleted: 6.6

Deleted: 6.13

Deleted: 6.12

Deleted: 6.5

Deleted: 6.14

Table 188: Asynchronous Request Header Structure

Asynchronous Request Batch Item		
Object	**REQUIRED in Message**	**Comment**
Batch Item	Yes	Structure, see 6.15
Operation	Yes	See 6.2
Unique Batch Item ID	No	REQUIRED if *Batch Count* > 1, see 6.4
Request Payload	Yes	Structure, contents depend on the Operation, see 4and 5
Message Extension	No	See 6.16

Deleted: 6.15

Deleted: 6.2

Deleted: 6.4

Deleted: 4

Deleted: 5

Deleted: 6.16

Table 189: Asynchronous Request Batch Item Structure

Asynchronous Response Header		
Object	**REQUIRED in Message**	**Comment**
Response Header	Yes	Structure
Protocol Version	Yes	See 6.1
Time Stamp	Yes	See 6.5
Batch Count	Yes	See 6.14

Deleted: 6.1

Deleted: 6.5

Deleted: 6.14

Table 190: Asynchronous Response Header Structure

Asynchronous Response Batch Item

Object	REQUIRED in Message	Comment
Batch Item	Yes	Structure, see 6.15
Operation	Yes, if not a failure	See 6.2
Unique Batch Item ID	No	REQUIRED if present in Request Batch Item, see 6.4
Result Status	Yes	See 6.9
Result Reason	Yes, if Result Status is Failure	Only required if Result Status is *Failure*, otherwise optional, see 6.10
Result Message	No	OPTIONAL if Result Status is not *Pending* or *Success*, see 6.11
Asynchronous Correlation Value	Yes	Only present if Result Status is *Pending*, see 6.8
Response Payload	Yes, if not a failure	Structure, contents depend on the Operation, see 4and 5
Message Extension	No	See 6.16

Table 191: Asynchronous Response Batch Item Structure

Deleted: 6.15

Deleted: 6.2

Deleted: 6.4

Deleted: 6.9

Deleted: present if

Deleted: No

Formatted: Font: Italic

Deleted: not *Pending* or *Success*

Deleted: 6.10

Deleted: Only present

Deleted: 6.11

Deleted: 6.8

Deleted: 4

Deleted: 5

Deleted: 6.16

8 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 KMIP Server SHALL support authentication as defined in **[KMIP-Prof]**.

9 Message Encoding

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

9.1 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 need to encode or decode protocol messages, and to provide optimal alignment for both 32-bit and 64-bit processors. Minimizing bandwidth over the transport mechanism is considered to be of lesser importance.

9.1.1 TTLV Encoding Fields

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

9.1.1.1 Item Tag

An Item Tag is a three-byte 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 SHALL contain either the value 42 in hex or the value 54 in hex as the high order (first) byte. Tags defined by this specification contain hex 42 in the first byte. Extensions, which are permitted, but are not defined in this specification, contain the value 54 hex in the first byte. A list of defined Item Tags is in Section 9.1.3.1.

Deleted: 9.1.3.1

9.1.1.2 Item Type

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

Data Type	Coded Value in Hex
Structure	01
Integer	02
Long Integer	03
Big Integer	04
Enumeration	05
Boolean	06
Text String	07
Byte String	08
Date-Time	09
Interval	0A

Table 192: Allowed Item Type Values

9.1.1.3 Item Length

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

Data Type	Length
Structure	Varies, multiple of 8
Integer	4
Long Integer	8
Big Integer	Varies, multiple of 8
Enumeration	4
Boolean	8
Text String	Varies
Byte String	Varies
Date-Time	8
Interval	4

Table 193: Allowed Item Length Values

If the Item Type is Structure, then the Item Length is the total length of all of the sub-items contained in the structure, including any padding. If the Item Type is Integer, Enumeration, Text String, Byte String, or Interval, then the Item Length is the number of bytes excluding the padding bytes. Text Strings and Byte Strings SHALL be padded with the minimal number of bytes following the Item Value to obtain a multiple of eight bytes. Integers, Enumerations, and Intervals SHALL be padded with four bytes following the Item Value.

9.1.1.4 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 four-byte long (32 bit) binary signed numbers in 2's complement notation, transmitted big-endian.
- Long Integers are encoded as eight-byte long (64 bit) binary signed numbers in 2's complement notation, transmitted big-endian.
- Big Integers are encoded as a sequence of eight-bit bytes, in two's complement notation, transmitted big-endian. If the length of the sequence is not a multiple of eight bytes, then Big Integers SHALL be padded with the minimal number of leading sign-extended bytes to make the length a multiple of eight bytes. These padding bytes are part of the Item Value and SHALL be counted in the Item Length.
- Enumerations are encoded as four-byte long (32 bit) binary unsigned numbers transmitted big-endian. Extensions, which are permitted, but are not defined in this specification, contain the value 8 hex in the first nibble of the first byte.
- Booleans are encoded as an eight-byte value that SHALL either contain the hex value 0000000000000000, indicating the Boolean value *False,* or the hex value 0000000000000001, transmitted big-endian, indicating the Boolean value *True.*

- Text Strings are sequences of bytes that encode character values according to the UTF-8 encoding standard. There SHALL NOT be null-termination at the end of such strings.
- Byte Strings are sequences of bytes containing individual unspecified eight-bit binary values, and are interpreted in the same sequence order.
- Date-Time values are POSIX Time values encoded as Long Integers. POSIX Time, as described in IEEE Standard 1003.1 **[IEEE1003-1]**, is the number of seconds since the Epoch (1970 Jan 1, 00:00:00 UTC), not counting leap seconds.
- Intervals are encoded as four-byte long (32 bit) binary unsigned numbers, transmitted big-endian. They have a resolution of one second.
- Structure Values are encoded as the concatenated encodings of the elements of the structure. All structures defined in this specification SHALL have all of their fields encoded in the order in which they appear in their respective structure descriptions.

9.1.2 Examples

These examples are assumed to be encoding a Protocol Object whose tag is 420020. The examples are shown as a sequence of bytes in hexadecimal notation:

- An Integer containing the decimal value 8:

```
42 00 20 | 02 | 00 00 00 04 | 00 00 00 08 00 00 00 00
```

- A Long Integer containing the decimal value 123456789000000000:

```
42 00 20 | 03 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00
```

- A Big Integer containing the decimal value 1234567890000000000000000000:

```
42 00 20 | 04 | 00 00 00 10 | 00 00 00 00 03 FD 35 EB 6B C2 DF 46 18 08
00 00
```

- An Enumeration with value 255:

```
42 00 20 | 05 | 00 00 00 04 | 00 00 00 FF 00 00 00 00
```

- A Boolean with the value *True*:

```
42 00 20 | 06 | 00 00 00 08 | 00 00 00 00 00 00 00 01
```

- A Text String with the value "Hello World":

```
42 00 20 | 07 | 00 00 00 0B | 48 65 6C 6C 6F 20 57 6F 72 6C 64 00 00 00
00 00
```

- A Byte String with the value { 0x01, 0x02, 0x03 }:

```
42 00 20 | 08 | 00 00 00 03 | 01 02 03 00 00 00 00 00
```

- A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:

```
42 00 20 | 09 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8
```

- An Interval, containing the value for 10 days:

```
42 00 20 | 0A | 00 00 00 04 | 00 0D 2F 00 00 00 00 00
```

- A Structure containing an Enumeration, value 254, followed by an Integer, value 255, having tags 420004 and 420005 respectively:

```
42 00 20 | 01 | 00 00 00 20 | 42 00 04 | 05 | 00 00 00 04 | 00 00 00 FE
00 00 00 00| 42 00 05 | 02 | 00 00 00 04 | 00 00 00 FF 00 00 00 00
```

9.1.3 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 is only able to be used for communication between parties that have pre-agreed understanding of the specific extensions.

9.1.3.1 Tags

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

Tag	
Object	**Tag Value**
(Unused)	000000 - 420000
Activation Date	420001
Application Data	420002
Application Namespace	420003
Application Specific Information	420004
Archive Date	420005
Asynchronous Correlation Value	420006
Asynchronous Indicator	420007
Attribute	420008
Attribute Index	420009
Attribute Name	42000A
Attribute Value	42000B
Authentication	42000C
Batch Count	42000D
Batch Error Continuation Option	42000E
Batch Item	42000F
Batch Order Option	420010
Block Cipher Mode	420011
Cancellation Result	420012
Certificate	420013
Certificate Identifier	420014
Certificate Issuer	420015
Certificate Issuer Alternative Name	420016
Certificate Issuer Distinguished Name	420017
Certificate Request	420018
Certificate Request Type	420019

Tag	
Object	**Tag Value**
Certificate Subject	42001A
Certificate Subject Alternative Name	42001B
Certificate Subject Distinguished Name	42001C
Certificate Type	42001D
Certificate Value	42001E
Common Template-Attribute	42001F
Compromise Date	420020
Compromise Occurrence Date	420021
Contact Information	420022
Credential	420023
Credential Type	420024
Credential Value	420025
Criticality Indicator	420026
CRT Coefficient	420027
Cryptographic Algorithm	420028
Cryptographic Domain Parameters	420029
Cryptographic Length	42002A
Cryptographic Parameters	42002B
Cryptographic Usage Mask	42002C
Custom Attribute	42002D
D	42002E
Deactivation Date	42002F
Derivation Data	420030
Derivation Method	420031
Derivation Parameters	420032
Destroy Date	420033
Digest	420034
Digest Value	420035
Encryption Key Information	420036
G	420037
Hashing Algorithm	420038
Initial Date	420039
Initialization Vector	42003A
Issuer	42003B

Tag	
Object	**Tag Value**
Iteration Count	42003C
IV/Counter/Nonce	42003D
J	42003E
Key	42003F
Key Block	420040
Key Compression Type	420041
Key Format Type	420042
Key Material	420043
Key Part Identifier	420044
Key Value	420045
Key Wrapping Data	420046
Key Wrapping Specification	420047
Last Change Date	420048
Lease Time	420049
Link	42004A
Link Type	42004B
Linked Object Identifier	42004C
MAC/Signature	42004D
MAC/Signature Key Information	42004E
Maximum Items	42004F
Maximum Response Size	420050
Message Extension	420051
Modulus	420052
Name	420053
Name Type	420054
Name Value	420055
Object Group	420056
Object Type	420057
Offset	420058
Opaque Data Type	420059
Opaque Data Value	42005A
Opaque Object	42005B
Operation	42005C
Operation Policy Name	42005D
P	42005E

Tag	
Object	**Tag Value**
Padding Method	42005F
Prime Exponent P	420060
Prime Exponent Q	420061
Prime Field Size	420062
Private Exponent	420063
Private Key	420064
Private Key Template-Attribute	420065
Private Key Unique Identifier	420066
Process Start Date	420067
Protect Stop Date	420068
Protocol Version	420069
Protocol Version Major	42006A
Protocol Version Minor	42006B
Public Exponent	42006C
Public Key	42006D
Public Key Template-Attribute	42006E
Public Key Unique Identifier	42006F
Put Function	420070
Q	420071
Q String	420072
Qlength	420073
Query Function	420074
Recommended Curve	420075
Replaced Unique Identifier	420076
Request Header	420077
Request Message	420078
Request Payload	420079
Response Header	42007A
Response Message	42007B
Response Payload	42007C
Result Message	42007D
Result Reason	42007E
Result Status	42007F
Revocation Message	420080
Revocation Reason	420081
Revocation Reason Code	420082

Tag	
Object	**Tag Value**
Role Type	420083
Salt	420084
Secret Data	420085
Secret Data Type	420086
Serial Number	420087
Server Information	420088
Split Key	420089
Split Key Method	42008A
Split Key Parts	42008B
Split Key Threshold	42008C
State	42008D
Storage Status Mask	42008E
Symmetric Key	42008F
Template	420090
Template-Attribute	420091
Time Stamp	420092
Unique Batch Item ID	420093
Unique Identifier	420094
Usage Limits	420095
Usage Limits Byte Count	420096
Usage Limits Object Count	420097
Usage Limits Total Bytes	420098
Usage Limits Total Objects	420099
Validity Date	42009A
Validity Indicator	42009B
Vendor Extension	42009C
Vendor Identification	42009D
Wrapping Method	42009E
X	42009F
Y	4200A0
(Reserved)	4200A1 – 42FFFF
(Unused)	430000 – 53FFFF
Extensions	540000 – 54FFFF
(Unused)	550000 - FFFFFF

Table 194: Tag Values

9.1.3.2 Enumerations

The following tables define the values for enumerated lists.

9.1.3.2.1 Credential Type Enumeration

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

Table 195: Credential Type Enumeration

9.1.3.2.2 Key Compression Type Enumeration

Key Compression Type	
Name	**Value**
EC Public Key Type Uncompressed	00000001
EC Public Key Type X9.62 Compressed Prime	00000002
EC Public Key Type X9,62 Compressed Char2	00000003
EC Public Key Type X9.62 Hybrid	00000004
Extensions	8XXXXXXX

Table 196: Key Compression Type Enumeration

9.1.3.2.3 Key Format Type Enumeration

Key Format Type	
Name	**Value**
Raw	00000001
Opaque	00000002
PKCS#1	00000003
PKCS#8	00000004
X.509	00000005
ECPrivateKey	00000006
Transparent Symmetric Key	00000007
Transparent DSA Private Key	00000008
Transparent DSA Public Key	00000009
Transparent RSA Private Key	0000000A

Transparent RSA Public Key	0000000B
Transparent DH Private Key	0000000C
Transparent DH Public Key	0000000D
Transparent ECDSA Private Key	0000000E
Transparent ECDSA Public Key	0000000F
Transparent ECDH Private Key	00000010
Transparent ECDH Public Key	00000011
Transparent ECMQV Private Key	00000012
Transparent ECMQV Public Key	00000013
Extensions	8XXXXXXX

Table 197: Key Format Type Enumeration

9.1.3.2.4 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
Extensions	8XXXXXXX

Table 198: Wrapping Method Enumeration

9.1.3.2.5 Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV

Recommended curves are defined in NIST FIPS PUB 186-3.

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`

Table 199: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV

9.1.3.2.6 Certificate Type Enumeration

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

Table 200: Certificate Type Enumeration

9.1.3.2.7 Split Key Method Enumeration

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

Table 201: Split Key Method Enumeration

9.1.3.2.8 Secret Data Type Enumeration

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

Table 202: Secret Data Type Enumeration

9.1.3.2.9 Opaque Data Type Enumeration

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

Table 203: Opaque Data Type Enumeration

9.1.3.2.10 Name Type Enumeration

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

Table 204: Name Type Enumeration

9.1.3.2.11 Object Type Enumeration

Object Type	
Name	**Value**
Certificate	00000001
Symmetric Key	00000002
Public Key	00000003
Private Key	00000004
Split Key	00000005
Template	00000006
Secret Data	00000007
Opaque Object	00000008
Extensions	8XXXXXXX

Table 205: Object Type Enumeration

1756 9.1.3.2.12 Cryptographic Algorithm Enumeration

Cryptographic Algorithm	
Name	**Value**
DES	00000001
3DES	00000002
AES	00000003
RSA	00000004
DSA	00000005
ECDSA	00000006
HMAC-SHA1	00000007
HMAC-SHA224	00000008
HMAC-SHA256	00000009
HMAC-SHA384	0000000A
HMAC-SHA512	0000000B
HMAC-MD5	0000000C
DH	0000000D
ECDH	0000000E
ECMQV	0000000F
Extensions	8XXXXXXX

1757 **Table 206: Cryptographic Algorithm Enumeration**

9.1.3.2.13 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
XTS	0000000B
AESKeyWrapPadding	0000000C
NISTKeyWrap	0000000D
X9.102 AESKW	0000000E
X9.102 TDKW	0000000F
X9.102 AKW1	00000010
X9.102 AKW2	00000011
Extensions	8XXXXXXX

Table 207: Block Cipher Mode Enumeration

9.1.3.2.14 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
X9.31	00000009
PSS	0000000A
Extensions	8XXXXXXX

Table 208: Padding Method Enumeration

9.1.3.2.15 Hashing Algorithm Enumeration

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

Table 209: Hashing Algorithm Enumeration

9.1.3.2.16 Role Type Enumeration

Role Type	
Name	**Value**
BDK	00000001
CVK	00000002
DEK	00000003
MKAC	00000004
MKSMC	00000005
MKSMI	00000006
MKDAC	00000007
MKDN	00000008
MKCP	00000009
MKOTH	0000000A
KEK	0000000B
MAC16609	0000000C
MAC97971	0000000D
MAC97972	0000000E
MAC97973	0000000F
MAC97974	00000010
MAC97975	00000011
ZPK	00000012
PVKIBM	00000013
PVKPVV	00000014
PVKOTH	00000015
Extensions	8XXXXXXX

Table 210: Role Type Enumeration

Note that while the set and definitions of role types are chosen to match TR-31 there is no necessity to match binary representations.

9.1.3.2.17 State Enumeration

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

Extensions	8XXXXXXX

Table 211: State Enumeration

9.1.3.2.18 Revocation Reason Code Enumeration

Revocation Reason Code	
Name	**Value**
Unspecified	00000001
Key Compromise	00000002
CA Compromise	00000003
Affiliation Changed	00000004
Superseded	00000005
Cessation of Operation	00000006
Privilege Withdrawn	00000007
Extensions	8XXXXXXX

Table 212: Revocation Reason Code Enumeration

9.1.3.2.19 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

Table 213: Link Type Enumeration

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

9.1.3.2.20 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

Table 214: Derivation Method Enumeration

9.1.3.2.21 Certificate Request Type Enumeration

Certificate Request Type	
Name	**Value**
CRMF	00000001
PKCS#10	00000002
PEM	00000003
PGP	00000004
Extensions	8XXXXXXX

Table 215: Certificate Request Type Enumeration

9.1.3.2.22 Validity Indicator Enumeration

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

Table 216: Validity Indicator Enumeration

9.1.3.2.23 Query Function Enumeration

Query Function	
Name	**Value**
Query Operations	00000001
Query Objects	00000002
Query Server Information	00000003

Query Application Namespaces	00000004
Extensions	8XXXXXXX

Table 217: Query Function Enumeration

9.1.3.2.24 Cancellation Result Enumeration

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

Table 218: Cancellation Result Enumeration

9.1.3.2.25 Put Function Enumeration

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

Table 219: Put Function Enumeration

9.1.3.2.26 Operation 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

Table 220: Operation Enumeration

9.1.3.2.27 Result Status Enumeration

Result Status	
Name	**Value**
Success	00000000
Operation Failed	00000001
Operation Pending	00000002
Operation Undone	00000003
Extensions	8XXXXXXX

Table 221: Result Status Enumeration

9.1.3.2.28 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
Index Out of Bounds	0000000E
Application Namespace Not Supported	0000000F
Key Format Type and/or Key Compression Type Not Supported	00000010
General Failure	00000100
Extensions	8XXXXXXX

Table 222: Result Reason Enumeration

9.1.3.2.29 Batch Error Continuation Enumeration

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

Table 223: Batch Error Continuation Enumeration

9.1.3.3 Bit Masks

9.1.3.3.1 Cryptographic Usage Mask

Cryptographic Usage Mask	
Name	**Value**
Sign	00000001
Verify	00000002
Encrypt	00000004
Decrypt	00000008
Wrap Key	00000010
Unwrap Key	00000020
Export	00000040
MAC Generate	00000080
MAC Verify	00000100
Derive Key	00000200
Content Commitment (Non Repudiation)	00000400
Key Agreement	00000800
Certificate Sign	00001000
CRL Sign	00002000
Generate Cryptogram	00004000
Validate Cryptogram	00008000
Translate Encrypt	00010000
Translate Decrypt	00020000
Translate Wrap	00040000
Translate Unwrap	00080000
Extensions	XXX00000

Table 224: Cryptographic Usage Mask

This list takes into consideration values which MAY appear in the Key Usage extension in an X.509 certificate.

9.1.3.3.2 Storage Status Mask

Storage Status Mask	
Name	**Value**
On-line storage	`00000001`
Archival storage	`00000002`
Extensions	`XXXXXXX0`

Table 225: Storage Status Mask

9.2 XML Encoding

An XML Encoding has not yet been defined.

10 Transport

A KMIP Server SHALL establish and maintain channel confidentiality and integrity, and provide assurance of server authenticity for KMIP messaging.

Deleted: prove

If a KMIP Server uses TCP/IP for KMIP messaging, then it SHALL support SSL v3.1/TLS v1.0 or later and may support other protocols as specified in **[KMIP-Prof]**.

11 Error Handling

This section details the specific Result Reasons that SHALL be returned for errors detected.

11.1 General

These errors MAY occur when any protocol message is received by the server or client (in response to server-to-client operations).

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	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; 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 is not able to 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

Table 226: General Errors

11.2 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	Operation Failed	Invalid Field
Error creating cryptographic object	Operation Failed	Cryptographic Failure
Trying to set more instances than the server supports of an attribute that MAY have multiple instances	Operation Failed	Index Out of Bounds
Trying to create a new object with the same Name attribute value as an existing object	Operation Failed	Invalid Field
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Template object is archived	Operation Failed	Object Archived

Comment: EBB: This column serves no useful purpose for the messages, since the content is always FAILURE

Comment: MBJ: proposed resolution: No change. This reflects the value of the tag.

Table 227: Create Errors

11.3 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	Operation Failed	Cryptographic Failure
Trying to create a new object with the same Name attribute value as an existing object	Operation Failed	Invalid Field
Trying to set more instances than the server supports of an attribute that MAY have multiple instances	Operation Failed	Index Out of Bounds
REQUIRED field(s) missing	Operation Failed	Invalid Message
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Template object is archived	Operation Failed	Object Archived

Table 228: Create Key Pair Errors

11.4 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	Operation Failed	Invalid Field
Trying to register a new object with the same Name attribute value as an existing object	Operation Failed	Invalid Field
Trying to set more instances than the server supports of an attribute that MAY have multiple instances	Operation Failed	Index Out of Bounds
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Template object is archived	Operation Failed	Object Archived

Table 229: Register Errors

11.5 Re-key

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object specified is not able to be re-keyed	Operation Failed	Permission Denied
Offset field is not permitted to 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
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived

Table 230: Re-key Errors

11.6 Derive Key

Error Definition	Result Status	Result Reason
One or more of the objects specified do not exist	Operation Failed	Item Not Found
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	Operation Failed	Invalid Field
One or more of the specified objects are not able to 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
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
One or more of the objects is archived	Operation Failed	Object Archived

Table 231: Derive Key Errors-

11.7 Certify

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object specified is not able to be certified	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
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported

Object is archived	Operation Failed	Object Archived

Table 232: Certify Errors

11.8 Re-certify

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object specified is not able to be certified	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 is not permitted to be specified at the same time as any of the Activation Date or Deactivation Date attributes	Operation Failed	Invalid Message
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived

Table 233: Re-certify Errors

11.9 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 the request	Operation Failed	Invalid Field

Table 234: Locate Errors

11.10 Check

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

Table 235: Check Errors

11.11 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 Encryption Key Information exists, but it is not a key	Operation Failed	Illegal Operation
Object with Encryption Key Information exists, but it is not able to be used for wrapping	Operation Failed	Permission Denied
Object with MAC/Signature Key Information exists, but it is not a key	Operation Failed	Illegal Operation
Object with MAC/Signature Key Information exists, but it is not able to be used for MACing/signing	Operation Failed	Permission Denied
Object exists but cannot be provided in the desired Key Format Type and/or Key Compression Type	Operation Failed	Key Format Type and/or Key Compression Type Not Supported
Object exists and is not a Template, but the server only has attributes for this object	Operation Failed	Illegal Operation
Cryptographic Parameters associated with the object do not exist or do not match those provided in the Encryption Key Information and/or Signature Key Information	Operation Failed	Item Not Found
Object is archived	Operation Failed	Object Archived

Deleted: Wrapping Key ID

Deleted: Wrapping Key ID

Deleted: ID

Deleted: ID

Table 236: Get Errors

11.12 Get Attributes

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
An Attribute Index is specified, but no matching instance exists.	Operation Failed	Item Not Found
Object is archived	Operation Failed	Object Archived

Table 237: Get Attributes Errors

11.13 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

Table 238: Get Attribute List Errors

11.14 Add Attribute

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Attempt to add a read-only attribute	Operation Failed	Permission Denied
Attempt to add an attribute that is not supported for this object	Operation Failed	Permission Denied
The specified attribute already exists	Operation Failed	Illegal Operation
New attribute contains Attribute Index	Operation Failed	Invalid Field
Trying to add a Name attribute with the same value that another object already has	Operation Failed	Illegal Operation
Trying to add a new instance to an attribute with multiple instances but the server limit on instances has been reached	Operation Failed	Index Out of Bounds
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived

Table 239: Add Attribute Errors

11.15 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 (i.e., it needs to 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 a value already used by another object	Operation Failed	Illegal Operation
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted	Operation Failed	Application Namespace Not Supported

from the client request		
Object is archived	Operation Failed	Object Archived

Table 240: Modify Attribute Errors

11.16 Delete Attribute

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Attempt to delete a read-only/REQUIRED attribute	Operation Failed	Permission Denied
Attribute Index is specified, but the attribute does not have multiple instances (i.e., no Attribute Index is permitted to be specified)	Operation Failed	Item Not Found
No attribute with the specified name exists	Operation Failed	Item Not Found
Object is archived	Operation Failed	Object Archived

Table 241: Delete Attribute Errors

11.17 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 is not permitted to be issued for the specified cryptographic object	Operation Failed	Permission Denied
Object is archived	Operation Failed	Object Archived

Table 242: Obtain Lease Errors

11.18 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 the object is not able to be used for applying cryptographic protection	Operation Failed	Illegal Operation
Both Usage Limits Byte Count and Usage Limits Object Count fields are specified	Operation Failed	Invalid Message
Neither the 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 MAY only be given for the other type	Operation Failed	Operation Not Supported
Object is archived	Operation Failed	Object Archived

Table 243: Get Usage Allocation Errors

11.19 Activate

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Unique Identifier specifies a template or other object that is not able to be activated	Operation Failed	Illegal Operation
Object is not in Pre-Active state	Operation Failed	Permission Denied
Object is archived	Operation Failed	Object Archived

Table 244: Activate Errors

11.20 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 a template or other object that is not able to be revoked	Operation Failed	Illegal Operation
Object is archived	Operation Failed	Object Archived

Table 245: Revoke Errors

11.21 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 Pre-Active, Deactivated or Compromised state	Operation Failed	Permission Denied
Object is archived	Operation Failed	Object Archived

Table 246: Destroy Errors

11.22 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

Table 247: Archive Errors

11.23 Recover

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

Table 248: Recover Errors

11.24 Validate

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

Table 249: Validate Errors

11.25 Query

N/A

11.26 Cancel

N/A

11.27 Poll

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

Table 250: Poll Errors

11.28 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 batch items was parsed correctly, then the response message SHOULD include response(s) to the batch item(s) in the request according to the table below.

Error Definition	Action	Result Reason
Processing of batch item fails with Batch Error Continuation Option set to Stop	Batch item fails and Result Status is set to Operation Failed. 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 and Result Status is set to Operation Failed. 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 and Result Status is set to Operation Failed. 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

Deleted: Result Status

Table 251: Batch Items Errors

12 Server Baseline Implementation Conformance Profile

The intention of the baseline conformance profile is for the minimal KMIP Server to support the mechanics of communication and to support a limited set of commands, such as query. The minimal KMIP Server would not need to support any particular algorithm – this would be the work of additional profiles.

An implementation is a conforming KMIP Server if the implementation meets the conditions in Section 12.1. **Deleted:** 12.1

An implementation SHALL be a conforming KMIP Server.

If an implementation claims support for a particular clause, then the implementation SHALL conform to all normative statements within that clause and any subclauses to that clause.

12.1 Conformance clauses for a KMIP Server

An implementation conforms to this specification as a KMIP Server if it meets the following conditions:

1. Supports the following objects:
   a. Attribute (see 2.1.1) **Deleted:** 2.1.1
   b. Credential (see 2.1.2) **Deleted:** 2.1.2
   c. Key Block (see 2.1.3) **Deleted:** 2.1.3
   d. Key Value (see 2.1.4) **Deleted:** 2.1.4
   e. Template-Attribute Structure (see 2.1.8) **Deleted:** 2.1.8
2. Supports the following attributes:
   a. Unique Identifier (see **Error! Reference source not found.**) **Deleted:** 3.1
   b. Name (see 3.2) **Deleted:** 3.2
   c. Object Type (see 3.3) **Deleted:** 3.3
   d. Cryptographic Algorithm (see 3.4) **Deleted:** 3.4
   e. Cryptographic Length (see 3.5) **Deleted:** 3.5
   f. Cryptographic Parameters (see 3.6) **Deleted:** 3.6
   g. Digest (see 3.12) **Deleted:** 3.12
   h. Default Operation Policy (see 3.13.2) **Deleted:** 3.13.2
   i. Cryptographic Usage Mask (see 3.14) **Deleted:** 3.14
   j. State (see 3.17) **Deleted:** 3.17
   k. Initial Date (see 3.18) **Deleted:** 3.18
   l. Activation Date (see 3.19) **Deleted:** 3.19
   m. Deactivation Date (see 3.22) **Deleted:** 3.22
   n. Destroy Date (see 3.23) **Deleted:** 3.23
   o. Compromise Occurrence Date (see 3.24) **Deleted:** 3.24
   p. Compromise Date (see 3.25) **Deleted:** 3.25
   q. Revocation Reason (see 3.26) **Deleted:** 3.26
   r. Archive Date (see 3.27) **Deleted:** 3.27
   s. Last Change Date (see 3.32) **Deleted:** 3.32
3. Supports the following client-to-server operations:

   a. Locate (see 4.8)
   b. Check (see 4.9)
   c. Get (see 4.10)
   d. Get Attribute (see 4.11)
   e. Get Attribute List (see 4.12)
   f. Add Attribute (see 4.13)
   g. Modify Attribute (see 4.14)
   h. Delete Attribute (see 4.15)
   i. Activate (see 4.18)
   j. Revoke (see 4.19)
   k. Destroy (see 4.20)
   l. Query (see 4.24)
4. Supports the following message contents:
   a. Protocol Version (see 6.1)
   b. Operation (see 6.2)
   c. Maximum Response Size (see 6.3)
   d. Unique Batch Item ID (see 6.4)
   e. Time Stamp (see 6.5)
   f. Asynchronous Indicator (see 6.7)
   g. Result Status (see 6.9)
   h. Result Reason (see 6.10)
   i. Result Message (see 6.11)
   j. Batch Order Option (see 6.12)
   k. Batch Error Continuation Option (see 6.13)
   l. Batch Count (see 6.14)
   m. Batch Item (see 6.15)
5. Supports Message Format (see 7)
6. Supports Authentication (see 8)
7. Supports the TTLV encoding (see 9.1)
8. Supports the transport requirements (see 10)
9. Supports Error Handling (see 11) for any supported object, attribute, or operation
10. Optionally supports any clause within this specification that is not listed above
11. Optionally supports extensions outside the scope of this standard (e.g., vendor extensions, conformance profiles) that do not contradict any requirements within this standard
12. Supports at least one of the profiles defined in the KMIP Profiles Specification **[KMIP-Prof]**.

Deleted: 4.8

Deleted: 4.9

Deleted: 4.10

Deleted: 4.11

Deleted: 4.12

Deleted: 4.13

Deleted: 4.14

Deleted: 4.15

Deleted: 4.18

Deleted: 4.19

Deleted: 4.20

Deleted: 4.24

Deleted: 6.1

Deleted: 6.2

Deleted: 6.3

Deleted: 6.4

Deleted: 6.5

Deleted: 6.7

Deleted: 6.9

Deleted: 6.10

Comment: EBB: Why would this feature be required?

Deleted: 6.11

Comment: MBJ: Proposed resolution: TBD.

Deleted: 6.12

Deleted: 6.13

Deleted: 6.14

Deleted: 6.15

Deleted: 7

Deleted: 8

Deleted: 9.1

Deleted: 10

Deleted: 11

A. Attribute Cross-reference

The following table of Attribute names indicates the Managed Object(s) for which each attribute applies. This table is not normative.

Comment: EBB: While this is correct, anything that doesn't have an X shouldn't be allowed

Comment: MBJ: proposed resolution: No change. This is not normative.

Attribute Name	Managed Object							
	Certificate	Symmetric Key	Public Key	Private Key	Split Key	Template	Secret Data	Opaque Object
Unique Identifier	x	x	x	x	x	x	x	x
Name	x	x	x	x	x	x	x	x
Object Type	x	x	x	x	x	x	x	x
Cryptographic Algorithm	x	x	x	x	x	x		
Cryptographic Domain Parameters			x	x		x		
Cryptographic Length	x	x	x	x	x	x		
Cryptographic Parameters	x	x	x	x	x	x		
Certificate Type	x							
Certificate Identifier	x							
Certificate Issuer	x							
Certificate Subject	x							
Digest	x	x	x	x	x		x	
Operation Policy Name	x	x	x	x	x	x	x	x
Cryptographic Usage Mask	x	x	x	x	x	x	x	
Lease Time	x	x	x	x	x		x	x
Usage Limits		x	x	x	x	x		
State	x	x	x	x	x		x	
Initial Date	x	x	x	x	x	x	x	x
Activation Date	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	x	x	x
Destroy Date	x	x	x	x	x		x	x
Compromise Occurrence Date	x	x	x	x	x		x	x
Compromise Date	x	x	x	x	x		x	x
Revocation Reason	x	x	x	x	x		x	x
Archive Date	x	x	x	x	x	x	x	x

	Managed Object							
Object Group	x	x	x	x	x	x	x	x
Link	x	x	x	x	x		x	
Application Specific Information	x	x	x	x	x	x	x	x
Contact Information	x	x	x	x	x	x	x	x
Last Change Date	x	x	x	x	x	x	x	x
Custom Attribute	x	x	x	x	x	x	x	x

Table 252: Attribute Cross-reference

B. Tag Cross-reference

This table is not normative.

Object	Defined	Type	Notes
Activation Date	3.19	Date-Time	
Application Data	3.30	Text String	
Application Namespace	3.30	Text String	
Application Specific Information	3.30	Structure	
Archive Date	3.27	Date-Time	
Asynchronous Correlation Value	6.8	Byte 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, 9.1.3.2.29	Enumeration	
Batch Item	6.15	Structure	
Batch Order Option	6.12	Boolean	
Block Cipher Mode	3.6, 9.1.3.2.13	Enumeration	
Cancellation Result	4.25, 9.1.3.2.24	Enumeration	
Certificate	2.2.1	Structure	
Certificate Identifier	3.9	Structure	
Certificate Issuer	3.9	Structure	
Certificate Issuer Alternative Name	3.11	Text String	
Certificate Issuer Distinguished Name	3.11	Text String	
Certificate Request	4.6, 4.7	Byte String	
Certificate Request Type	4.6, 4.7, 9.1.3.2.21	Enumeration	
Certificate Subject	3.10	Structure	
Certificate Subject Alternative Name	3.10	Text String	
Certificate Subject Distinguished Name	3.10	Text String	
Certificate Type	2.2.1, 3.8 , 9.1.3.2.6	Enumeration	
Certificate Value	2.2.1	Byte String	
Common Template-Attribute	2.1.8	Structure	
Compromise Occurrence Date	3.24	Date-Time	
Compromise Date	3.25	Date-Time	
Contact Information	3.31	Text String	

Deleted: 3.19

Deleted: 3.30

Deleted: 3.30

Deleted: 3.30

Deleted: 3.27

Deleted: 6.8

Deleted: 6.7

Deleted: 2.1.1

Deleted: 2.1.1

Deleted: 2.1.1

Deleted: 2.1.1

Deleted: 6.6

Deleted: 6.14

Deleted: 6.13

Deleted: 9.1.3.2.29

Deleted: 6.15

Deleted: 6.12

Deleted: 3.6

Deleted: 9.1.3.2.13

Deleted: 4.25

Deleted: 9.1.3.2.24

Deleted: 2.2.1

Deleted: 3.9

Deleted: 3.9

Deleted: 3.11

Deleted: 3.11

Deleted: 4.6

Deleted: 4.7

Deleted: 4.6

Deleted: 4.7

Deleted: 9.1.3.2.21

Deleted: 3.10

Deleted: 3.10

Deleted: 3.10

Deleted: 2.2.1

Deleted: 3.8

Deleted: 9.1.3.2.6

Deleted: 2.2.1

Deleted: 2.1.8

Deleted: 3.24

Deleted: 3.25

Deleted: 3.31

Object	Defined	Type	Notes
Credential	2.1.2	Structure	
Credential Type	2.1.2, 9.1.3.2.1	Enumeration	
Credential Value	2.1.2	Byte String	
Criticality Indicator	6.16	Boolean	
CRT Coefficient	2.1.7	Big Integer	
Cryptographic Algorithm	3.4, 9.1.3.2.12	Enumeration	
Cryptographic Length	3.5	Integer	
Cryptographic Parameters	3.6	Structure	
Cryptographic Usage Mask	3.14, 9.1.3.3.1	Integer	Bit mask
Custom Attribute	3.33	*	type varies
D	2.1.7	Big Integer	
Deactivation Date	3.22	Date-Time	
Derivation Data	4.5	Byte String	
Derivation Method	4.5, 9.1.3.2.20	Enumeration	
Derivation Parameters	4.5	Structure	
Destroy Date	3.23	Date-Time	
Digest	3.12	Structure	
Digest Value	3.12	Byte String	
Encryption Key Information	2.1.5	Structure	
Extensions	9.1.3		
G	2.1.7	Big Integer	
Hashing Algorithm	3.6, 3.12, 9.1.3.2.15	Enumeration	
Initial Date	3.18	Date-Time	
Initialization Vector	4.5	Byte String	
Issuer	3.9	Text String	
Iteration Count	4.5	Integer	
IV/Counter/Nonce	2.1.5	Byte String	
J	2.1.7	Big Integer	
Key	2.1.7	Byte String	
Key Block	2.1.3	Structure	
Key Compression Type	9.1.3.2.2	Enumeration	
Key Format Type	2.1.4, 9.1.3.2.3	Enumeration	
Key Material	2.1.4, 2.1.7	Byte String / Structure	
Key Part Identifier	2.2.5	Integer	
Key Value	2.1.4	Byte String / Structure	
Key Wrapping Data	2.1.5	Structure	
Key Wrapping Specification	2.1.6	Structure	

Deleted: 2.1.2

Deleted: 2.1.2 …9.1.3.2.1 ... [2]

Deleted: 2.1.2

Deleted: 6.16

Deleted: 2.1.7

Deleted: 3.4 …9.1.3.2.12 ... [3]

Deleted: 3.5

Deleted: 3.6

Deleted: 3.14 …9.1.3.3.1 ... [4]

Deleted: 3.33

Deleted: 2.1.7

Deleted: 3.22

Deleted: 4.5

Deleted: 4.5 …9.1.3.2.20 ... [5]

Deleted: 4.5

Deleted: 3.23

Deleted: 3.12

Deleted: 3.12

Deleted: 2.1.5

Deleted: 9.1.3

Deleted: 2.1.7

Deleted: 3.6 …3.12 …9.1.3.2. ... [6]

Deleted: 3.18

Deleted: 4.5

Deleted: 3.9

Deleted: 4.5

Deleted: 2.1.5

Deleted: 2.1.7

Deleted: 2.1.7

Deleted: 2.1.3

Deleted: 9.1.3.2.2

Deleted: 2.1.4 …9.1.3.2.3 ... [7]

Deleted: 2.1.4 …2.1.7 ... [8]

Deleted: 2.2.5

Deleted: 2.1.4

Deleted: 2.1.5

Deleted: 2.1.6

Object	Defined	Type	Notes
Last Change Date	3.32	Date-Time	
Lease Time	3.15	Interval	
Link	3.29	Structure	
Link Type	3.29, 9.1.3.2.19	Enumeration	
Linked Object Identifier	3.29	Text String	
MAC/Signature	2.1.5	Byte 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, 9.1.3.2.10	Enumeration	
Name Value	3.2	Text String	
Object Group	3.28	Text String	
Object Type	3.3, 9.1.3.2.11	Enumeration	
Offset	4.4, 4.7	Interval	
Opaque Data Type	2.2.8, 9.1.3.2.9	Enumeration	
Opaque Data Value	2.2.8	Byte String	
Opaque Object	2.2.8	Structure	
Operation	6.2, 9.1.3.2.26	Enumeration	
Operation Policy Name	3.13	Text String	
P	2.1.7	Big Integer	
Padding Method	3.6, 9.1.3.2.14	Enumeration	
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	3.20	Date-Time	
Protect Stop Date	3.21	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	

Deleted: 3.32

Deleted: 3.15

Deleted: 3.29

Deleted: 3.29 ...9.1.3.2.19 ... [9]

Deleted: 3.29

Deleted: 2.1.5

Deleted: 2.1.5

Deleted: 4.8

Deleted: 6.3

Deleted: 6.16

Deleted: 2.1.7

Deleted: 3.2

Deleted: 3.2 ...9.1.3.2.10 ... [10]

Deleted: 3.2

Deleted: 3.28

Deleted: 3.3 ...9.1.3.2.11 ... [11]

Deleted: 4.4 ...4.7 ... [12]

Deleted: 2.2.8 ...9.1.3.2.9 ... [13]

Deleted: 2.2.8

Deleted: 2.2.8

Deleted: 6.2 ...9.1.3.2.26 ... [14]

Deleted: 3.13

Deleted: 2.1.7

Deleted: 3.6 ...9.1.3.2.14 ... [15]

Deleted: 2.1.7

Deleted: 2.1.7

Deleted: 2.2.5

Deleted: 2.1.7

Deleted: 2.2.4

Deleted: 2.1.8

Deleted: 4.2

Deleted: 3.20

Deleted: 3.21

Deleted: 6.1

Deleted: 6.1

Deleted: 6.1

Deleted: 2.1.7

Deleted: 2.2.3

Object	Defined	Type	Notes
Public Key Template-Attribute	2.1.8	Structure	
Public Key Unique Identifier	4.2	Text String	
Put Function	5.2, 9.1.3.2.25	Enumeration	
Q	2.1.7	Big Integer	
Q String	2.1.7	Byte String	
Qlength	3.7	Integer	
Query Function	4.24, 9.1.3.2.23	Enumeration	
Recommended Curve	2.1.7, 3.7, 9.1.3.2.5	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, 9.1.3.2.28	Enumeration	
Result Status	6.9, 9.1.3.2.27	Enumeration	
Revocation Message	3.26	Text String	
Revocation Reason	3.26	Structure	
Revocation Reason Code	3.26, 9.1.3.2.18	Enumeration	
Role Type	3.6, 9.1.3.2.16	Enumeration	
Salt	4.5	Byte String	
Secret Data	2.2.7	Structure	
Secret Data Type	2.2.7, 9.1.3.2.8	Enumeration	
Serial Number	3.9	Text String	
Server Information	4.24	Structure	contents vendor-specific
Split Key	2.2.5	Structure	
Split Key Method	2.2.5, 9.1.3.2.7	Enumeration	
Split Key Parts	2.2.5	Integer	
Split Key Threshold	2.2.5	Integer	
State	3.17, 9.1.3.2.17	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-Attribute	2.1.8	Structure	
Time Stamp	6.5	Date-Time	
Transparent*	2.1.7	Structure	

Deleted: 2.1.8

Deleted: 4.2

Deleted: 5.2 …9.1.3.2.25 ... [16]

Deleted: 2.1.7

Deleted: 2.1.7

Deleted: 3.7

Deleted: 4.24 …9.1.3.2.23 ... [17]

Deleted: 2.1.7 …3.7 …9.1.3. ... [18]

Deleted: 5.2

Deleted: 7.2 …7.3 ... [19]

Deleted: 7.1

Deleted: 4 …5 …7.2 …7.3 ... [20]

Deleted: 7.2 …7.3 ... [21]

Deleted: 7.1

Deleted: 4 …7.2 …7.3 ... [22]

Deleted: 6.11

Deleted: 6.10 …9.1.3.2.28 ... [23]

Deleted: 6.9 …9.1.3.2.27 ... [24]

Deleted: 3.26

Deleted: 3.26

Deleted: 3.26 …9.1.3.2.18 ... [25]

Deleted: 3.6 …9.1.3.2.16 ... [26]

Deleted: 4.5

Deleted: 2.2.7

Deleted: 2.2.7 …9.1.3.2.8 ... [27]

Deleted: 3.9

Deleted: 4.24

Deleted: 2.2.5

Deleted: 2.2.5 …9.1.3.2.7 ... [28]

Deleted: 2.2.5

Deleted: 2.2.5

Deleted: 3.17 …9.1.3.2.17 ... [29]

Deleted: 4.8 …9.1.3.3.2 ... [30]

Deleted: 2.2.2

Deleted: 2.2.6

Deleted: 2.1.8

Deleted: 6.5

Deleted: 2.1.7

Object	Defined	Type	Notes
Unique Identifier	3.1	Text String	
Unique Batch Item ID	6.4	Byte String	
Usage Limits	3.16	Structure	
Usage Limits Byte Count	3.16	Big Integer	
Usage Limits Object Count	3.16	Big Integer	
Usage Limits Total Bytes	3.16	Big Integer	
Usage Limits Total Objects	3.16	Big Integer	
Validity Date	4.23	Date-Time	
Validity Indicator	4.23, 9.1.3.2.22	Enumeration	
Vendor Extension	6.16	Structure	contents vendor-specific
Vendor Identification	4.24, 6.16	Text String	
Wrapping Method	2.1.5, 9.1.3.2.4	Enumeration	
X	2.1.7	Big Integer	
Y	2.1.7	Big Integer	

Table 253: Tag Cross-reference

Deleted: 3.1

Deleted: 6.4

Deleted: 3.16

Deleted: 3.16

Deleted: 3.16

Deleted: 3.16

Deleted: 3.16

Deleted: 4.23

Deleted: 4.23

Deleted: 9.1.3.2.22

Deleted: 6.16

Deleted: 4.24

Deleted: 6.16

Deleted: 2.1.5

Deleted: 9.1.3.2.4

Deleted: 2.1.7

Deleted: 2.1.7

C. Operation and Object Cross-reference

The following table indicates the types of Managed Object(s) that each Operation accepts as input or provides as output. This table is not normative.

Deleted: provide

Operation	Managed Objects							
	Certificate	Symmetric Key	Public Key	Private Key	Split Key	Template	Secret Data	Opaque Object
Create	N/A	Y	N/A	N/A	N/A	Y	N/A	N/A
Create Key Pair	N/A	N/A	Y	Y	N/A	N/A	N/A	N/A
Register	Y	Y	Y	Y	Y	Y	Y	Y
Re-Key	N/A	Y	N/A	N/A	N/A	Y	N/A	N/A
Derive Key	N/A	Y	N/A	N/A	N/A	Y	Y	N/A
Certify	Y	N/A	Y	N/A	N/A	Y	N/A	N/A
Re-certify	Y	N/A	N/A	N/A	N/A	Y	N/A	N/A
Locate	Y	Y	Y	Y	Y	Y	Y	Y
Check	Y	Y	Y	Y	Y	N/A	Y	Y
Get	Y	Y	Y	Y	Y	Y	Y	Y
Get Attributes	Y	Y	Y	Y	Y	Y	Y	Y
Get Attribute List	Y	Y	Y	Y	Y	Y	Y	Y
Add Attribute	Y	Y	Y	Y	Y	Y	Y	Y
Modify Attribute	Y	Y	Y	Y	Y	Y	Y	Y
Delete Attribute	Y	Y	Y	Y	Y	Y	Y	Y
Obtain Lease	Y	Y	Y	Y	Y	N/A	Y	N/A
Get Usage Allocation	N/A	Y	Y	Y	N/A	N/A	N/A	N/A
Activate	Y	Y	Y	Y	Y	N/A	Y	N/A
Revoke	Y	Y	N/A	Y	Y	N/A	Y	Y
Destroy	Y	Y	Y	Y	Y	Y	Y	Y
Archive	Y	Y	Y	Y	Y	Y	Y	Y
Recover	Y	Y	Y	Y	Y	Y	Y	Y
Validate	Y	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Query	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Cancel	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Poll	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Notify	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Put	Y	Y	Y	Y	Y	Y	Y	Y

Table 254: Operation and Object Cross-reference

D. Acronyms

The following abbreviations and acronyms are used in this document:

3DES	- Triple Data Encryption Standard specified in ANSI X9.52
AES	- Advanced Encryption Standard specified in FIPS 197
ASN.1	- Abstract Syntax Notation One specified in ITU-T X.680
BDK	- Base Derivation Key specified in ANSI X9 TR-31
CA	- Certification Authority
CBC	- Cipher Block Chaining
CCM	- Counter with CBC-MAC specified in NIST SP 800-38C
CFB	- Cipher Feedback specified in NIST SP 800-38A
CMAC	- Cipher-based MAC specified in NIST SP 800-38B
CMC	- Certificate Management Messages over CMS specified in RFC 5275
CMP	- Certificate Management Protocol specified in RFC 4210
CPU	- Central Processing Unit
CRL	- Certificate Revocation List specified in RFC 5280
CRMF	- Certificate Request Message Format specified in RFC 4211
CRT	- Chinese Remainder Theorem
CTR	- Counter specified in NIST SP 800-38A
CVK	- Card Verification Key specified in ANSI X9 TR-31
DEK	- Data Encryption Key
DER	- Distinguished Encoding Rules specified in ITU-T X.690
DES	- Data Encryption Standard specified in FIPS 46-3
DH	- Diffie-Hellman specified in ANSI X9.42
DNS	- Domain Name Server
DSA	- Digital Signature Algorithm specified in FIPS 186-3
DSKPP	- Dynamic Symmetric Key Provisioning Protocol
ECB	- Electronic Code Book
ECDH	- Elliptic Curve Diffie-Hellman specified in ANSI X9.63 and NIST SP 800-56A
ECDSA	- Elliptic Curve Digital Signature Algorithm specified in ANSX9.62
ECMQV	- Elliptic Curve Menezes Qu Vanstone specified in ANSI X9.63 and NIST SP 800-56A
FIPS	- Federal Information Processing Standard
GCM	- Galois/Counter Mode specified in NIST SP 800-38D
GF	- Galois field (or finite field)
HMAC	- Keyed-Hash Message Authentication Code specified in FIPS 198-1 and RFC 2104
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
IP	- Internet Protocol
IPsec	- Internet Protocol Security
IV	- Initialization Vector
KEK	- Key Encryption Key
KMIP	- Key Management Interoperability Protocol
MAC	- Message Authentication Code
MKAC	- EMV/chip card Master Key: Application Cryptograms specified in ANSI X9 TR-31
MKCP	- EMV/chip card Master Key: Card Personalization specified in ANSI X9 TR-31
MKDAC	- EMV/chip card Master Key: Data Authentication Code specified in ANSI X9 TR-31
MKDN	- EMV/chip card Master Key: Dynamic Numbers specified in ANSI X9 TR-31
MKOTH	- EMV/chip card Master Key: Other specified in ANSI X9 TR-31
MKSMC	- EMV/chip card Master Key: Secure Messaging for Confidentiality specified in X9 TR-31
MKSMI	- EMV/chip card Master Key: Secure Messaging for Integrity specified in ANSI X9 TR-31
MD2	- Message Digest 2 Algorithm specified in RFC 1319
MD4	- Message Digest 4 Algorithm specified in RFC 1320
MD5	- Message Digest 5 Algorithm specified in RFC 1321
NIST	- National Institute of Standards and Technology
OAEP	- Optimal Asymmetric Encryption Padding specified in PKCS#1
OFB	- Output Feedback specified in NIST SP 800-38A
PBKDF2	- Password-Based Key Derivation Function 2 specified in RFC 2898
PCBC	- Propagating Cipher Block Chaining
PEM	- Privacy Enhanced Mail specified in RFC 1421
PGP	- Pretty Good Privacy specified in RFC 1991
PKCS	- Public-Key Cryptography Standards
PKCS#1	- RSA Cryptography Specification Version 2.1 specified in RFC 3447
PKCS#5	- Password-Based Cryptography Specification Version 2 specified in RFC 2898
PKCS#8	- Private-Key Information Syntax Specification Version 1.2 specified in RFC 5208
PKCS#10	- Certification Request Syntax Specification Version 1.7 specified in RFC 2986
POSIX	- Portable Operating System Interface
RFC	- Request for Comments documents of IETF
RSA	- Rivest, Shamir, Adelman (an algorithm)
SCEP	- Simple Certificate Enrollment Protocol
SHA	- Secure Hash Algorithm specified in FIPS 180-2
SP	- Special Publication
SSL/TLS	- Secure Sockets Layer/Transport Layer Security
S/MIME	- Secure/Multipurpose Internet Mail Extensions
TDEA	- see 3DES

TCP	- Transport Control Protocol
TTLV	- Tag, Type, Length, Value
URI	- Uniform Resource Identifier
UTC	- Universal Time Coordinated
UTF	- Universal Transformation Format 8-bit specified in RFC 3629
XKMS	- XML Key Management Specification
XML	- Extensible Markup Language
XTS	- XEX Tweakable Block Cipher with Ciphertext Stealing specified in NIST SP 800-38E
X.509	- Public Key Certificate specified in RFC 5280
ZPK	- PIN Block Encryption Key specified in ANSI X9 TR-31

E. List of Figures and Tables

Deleted: 43

F. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Original Authors of the initial contribution:

David Babcock, HP
Steven Bade, IBM
Paolo Bezoari, NetApp
Mathias Björkqvist, IBM
Bruce Brinson, EMC
Christian Cachin, IBM
Tony Crossman, Thales/nCipher
Stan Feather, HP
Indra Fitzgerald, HP
Judy Furlong, EMC
Jon Geater, Thales/nCipher
Bob Griffin, EMC
Robert Haas, IBM (editor)
Timothy Hahn, IBM
Jack Harwood, EMC
Walt Hubis, LSI
Glen Jaquette, IBM
Jeff Kravitz, IBM (editor emeritus)
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, Thales/nCipher
John Tattan, EMC
Karla Thomas, Brocade
Marko Vukolić, IBM
Steve Wierenga, HP

Participants:

Gordon Arnold, IBM
Todd Arnold, IBM
Matthew Ball, Sun Microsystems
Elaine Barker, NIST
Peter Bartok, Venafi, Inc.
Mathias Björkqvist, IBM
Kevin Bocek, Thales e-Security
Kelley Burgin, National Security Agency
Jon Callas, PGP Corporation
Tom Clifford, Symantec Corp.
Graydon Dodson, Lexmark International Inc.
Chris Dunn, SafeNet, Inc.
Paul Earsy, SafeNet, Inc.
Stan Feather, HP
Indra Fitzgerald, HP
Alan Frindell, SafeNet, Inc.

Judith Furlong, EMC Corporation
Jonathan Geater, Thales e-Security
Robert Griffin, EMC Corporation
Robert Haas, IBM
Thomas Hardjono, M.I.T.
Marc Hocking, BeCrypt Ltd.
Larry Hofer, Emulex Corporation
Brandon Hoff, Emulex Corporation
Walt Hubis, LSI Corporation
Wyllys Ingersoll, Sun Microsystems
Jay Jacobs, Target Corporation
Glen Jaquette, IBM
Scott Kipp, Brocade Communications Systems, Inc.
David Lawson, Emulex Corporation
Robert Lockhart, Thales e-Security
Shyam Mankala, EMC Corporation
Marc Massar, Individual
Don McAlister, Associate
Hyrum Mills, Mitre Corporation
Landon Noll, Cisco Systems, Inc.
René Pawlitzek, IBM
Rob Philpott, EMC Corporation
Bruce Rich, IBM
Scott Rotondo, Sun Microsystems
Anil Saldhana, Red Hat
Subhash Sankuratripati, NetApp
Mark Schiller, HP
Jitendra Singh, Brocade Communications Systems, Inc.
Servesh Singh, EMC Corporation
Sandy Stewart, Sun Microsystems
Marcus Streets, Thales e-Security
Brett Thompson, SafeNet, Inc.
Benjamin Tomhave, Individual
Sean Turner, IECA, Inc.
Paul Turner, Venafi, Inc.
Marko Vukolic, IBM
Rod Wideman, Quantum Corporation
Steven Wierenga, HP
Peter Yee, EMC Corporation
Krishna Yellepeddy, IBM
Peter Zelechoski, Associate

2396 # G. Revision History

Revision	Date	Editor	Changes Made
ed-0.98	2009-04-24	Robert Haas	Initial conversion of input document to OASIS format together with clarifications.
ed-0.98	2009-05-21	Robert Haas	Changes to TTLV format for 64-bit alignment. Appendices indicated as non normative.
ed-0.98	2009-06-25	Robert Haas, Indra Fitzgerald	Multiple editorial and technical changes, including merge of Template and Policy Template.
ed-0.98	2009-07-23	Robert Haas, Indra Fitzgerald	Multiple editorial and technical changes, mainly based on comments from Elaine Barker and Judy Furlong. Fix of Template Name.
ed-0.98	2009-07-27	Indra Fitzgerald	Added captions to tables and figures.
ed-0.98	2009-08-27	Robert Haas	Wording compliance changes according to RFC2119 from Rod Wideman. Removal of attribute mutation in server responses.
ed-0.98	2009-09-03	Robert Haas	Incorporated the RFC2119 language conformance statement from Matt Ball; the changes to the Application-Specific Information attribute from René Pawlitzek; the extensions to the Query operation for namespaces from Mathias Björkqvist; the key roles proposal from Jon Geater, Todd Arnold, & Chris Dunn. Capitalized all RFC2119 keywords (required by OASIS) together with editorial changes.
ed-0.98	2009-09-17	Robert Haas	Replaced Section 10 on HTTPS and SSL with the content from the User Guide. Additional RFC2119 language conformance changes. Corrections in the enumerations in Section 9.
ed-0.98	2009-09-25	Indra Fitzgerald, Robert Haas	New Cryptographic Domain Parameters attribute and change to the Create Key Pair operation (from Indra Fitzgerald). Changes to Key Block object and Get operation to request desired Key Format and Compression Types (from Indra Fitzgerald). Changes in Revocation Reason code and new Certificate Issuer attribute (from Judy Furlong). No implicit object state change after Re-key or Re-certify. New Section 13 on Implementation Conformance from Matt Ball. Multiple editorial changes and new enumerations.
ed-0.98	2009-09-29	Robert Haas	(Version edited during the f2f) Moved content of Sections 8 (Authentication) and 10 (Transport), into the KMIP Profiles Specification. Clarifications (from Sean Turner) on key encoding (for Byte String) in 9.1.1.4. Updates for certificate update and renewal (From Judy

			Furlong) First set of editorial changes as suggested by Elaine Barker (changed Octet to Byte, etc). (version approved as TC Committee Draft on Sep 29 2009, counts as draft-01 version)
draft-02	2009-10-09	Robert Haas, Indra Fitzgerald	Second set of editorial changes as suggested by Elaine Barker (incl. renaming of "Last Change Date" attribute). Added list of references from Sean Turner and Judy Furlong, as well as terminology. Made Result Reasons in error cases (Sec 11) normative. Added statement on deletion of attributes by server (line 457). Added major/minor 1.0 for protocol version (line 27). Systematic use of *italics* when introducing a term for first time. Added "Editor's note" comments remaining to be addressed before public review.
draft-03	2009-10-14	Robert Haas, Indra Fitzgerald	Addressed outstanding "Editor's note" comments. Added acronyms and references.
draft-04	2009-10-21	Robert Haas, Indra Fitzgerald	Added the list of participants (Appendix F). Point to the KMIP Profiles document for a list standard application namespaces. Added Terminology (from Bob Lockhart, borrowed from SP800-57 Part 1). Modified title page.
draft-05	2009-11-06	Robert Haas	Additions to the tags table. Added Last Change Date attribute to conformance clause (sec 12.1). Minor edits. This is the tentative revision for public review.
draft-06	2009-11-09	Robert Haas	Editorial fixes to the reference sections. Correction of the comments for the Unique Batch Item ID in the Response Header structures (from Steve Wierenga). Version used for Public Review 01.
draft-07	2010-02-04	Robert Haas	Editorial fixes according to Elaine Barker's comments. Comments for which the proposed resolution is "No Change" are indicated accordingly. Open issues marked with "TBD" and possible Usage Guide items are marked with "UG".

Page 90: [1] Formatted rha 2/4/2010 2:06 PM

Font: 10 pt, Check spelling and grammar

Page 135: [2] Deleted rha 2/4/2010 2:06 PM

2.1.2

Page 135: [2] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.1

Page 135: [3] Deleted rha 2/4/2010 2:06 PM

3.4

Page 135: [3] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.12

Page 135: [4] Deleted rha 2/4/2010 2:06 PM

3.14

Page 135: [4] Deleted rha 2/4/2010 2:06 PM

9.1.3.3.1

Page 135: [5] Deleted rha 2/4/2010 2:06 PM

4.5

Page 135: [5] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.20

Page 135: [6] Deleted rha 2/4/2010 2:06 PM

3.6

Page 135: [6] Deleted rha 2/4/2010 2:06 PM

3.12

Page 135: [6] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.15

Page 135: [7] Deleted rha 2/4/2010 2:06 PM

2.1.4

Page 135: [7] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.3

Page 135: [8] Deleted rha 2/4/2010 2:06 PM

2.1.4

Page 135: [8] Deleted rha 2/4/2010 2:06 PM

2.1.7

Page 136: [9] Deleted rha 2/4/2010 2:06 PM

3.29

Page 136: [9] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.19

Page 136: [10] Deleted rha 2/4/2010 2:06 PM

3.2

Page 136: [10] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.10

Page 136: [11] Deleted rha 2/4/2010 2:06 PM

3.3

Page 136: [11] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.11

Page 136: [12] Deleted rha 2/4/2010 2:06 PM

4.4

Page 136: [12] Deleted rha 2/4/2010 2:06 PM

4.7

Page 136: [13] Deleted rha 2/4/2010 2:06 PM

2.2.8

Page 136: [13] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.9

Page 136: [14] Deleted rha 2/4/2010 2:06 PM

6.2

Page 136: [14] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.26

Page 136: [15] Deleted rha 2/4/2010 2:06 PM

3.6

Page 136: [15] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.14

Page 137: [16] Deleted rha 2/4/2010 2:06 PM

5.2

Page 137: [16] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.25

Page 137: [17] Deleted rha 2/4/2010 2:06 PM

4.24

Page 137: [17] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.23

Page 137: [18] Deleted rha 2/4/2010 2:06 PM

2.1.7

Page 137: [18] Deleted rha 2/4/2010 2:06 PM

3.7

Page 137: [18] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.5

Page 137: [19] Deleted rha 2/4/2010 2:06 PM

7.2

Page 137: [19] Deleted rha 2/4/2010 2:06 PM

7.3

Page 137: [20] Deleted rha 2/4/2010 2:06 PM

4

Page 137: [20] Deleted rha 2/4/2010 2:06 PM

5

Page 137: [20] Deleted rha 2/4/2010 2:06 PM

7.2

Page 137: [20] Deleted rha 2/4/2010 2:06 PM

7.3

Page 137: [21] Deleted rha 2/4/2010 2:06 PM

7.2

Page 137: [21] Deleted rha 2/4/2010 2:06 PM

7.3

Page 137: [22] Deleted rha 2/4/2010 2:06 PM

4

Page 137: [22] Deleted rha 2/4/2010 2:06 PM

7.2

Page 137: [22] Deleted rha 2/4/2010 2:06 PM

7.3

Page 137: [23] Deleted rha 2/4/2010 2:06 PM

6.10

Page 137: [23] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.28

Page 137: [24] Deleted rha 2/4/2010 2:06 PM

6.9

Page 137: [24] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.27

Page 137: [25] Deleted rha 2/4/2010 2:06 PM

3.26

Page 137: [25] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.18

Page 137: [26] Deleted rha 2/4/2010 2:06 PM

3.6

Page 137: [26] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.16

Page 137: [27] Deleted rha 2/4/2010 2:06 PM

2.2.7

Page 137: [27] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.8

Page 137: [28] Deleted rha 2/4/2010 2:06 PM

2.2.5

Page 137: [28] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.7

Page 137: [29] Deleted rha 2/4/2010 2:06 PM

3.17

Page 137: [29] Deleted rha 2/4/2010 2:06 PM

9.1.3.2.17

Page 137: [30] Deleted rha 2/4/2010 2:06 PM

4.8

Page 137: [30] Deleted rha 2/4/2010 2:06 PM

9.1.3.3.2