Key Management Interoperability Protocol Specification Version 1.2

Committee Specification Draft 02 /
Public Review Draft 02

19 June 2014

Specification URIs

This version:

http://docs.oasis-open.org/kmip/spec/v1.2/csprd02/kmip-spec-v1.2-csprd02.doc (Authoritative)
http://docs.oasis-open.org/kmip/spec/v1.2/csprd02/kmip-spec-v1.2-csprd02.html
http://docs.oasis-open.org/kmip/spec/v1.2/csprd02/kmip-spec-v1.2-csprd02.pdf

Previous version:

http://docs.oasis-open.org/kmip/spec/v1.2/csprd01/kmip-spec-v1.2-csprd01.doc (Authoritative)
http://docs.oasis-open.org/kmip/spec/v1.2/csprd01/kmip-spec-v1.2-csprd01.html
http://docs.oasis-open.org/kmip/spec/v1.2/csprd01/kmip-spec-v1.2-csprd01.pdf

Latest version:

http://docs.oasis-open.org/kmip/spec/v1.2/kmip-spec-v1.2.doc (Authoritative)
http://docs.oasis-open.org/kmip/spec/v1.2/kmip-spec-v1.2.html
http://docs.oasis-open.org/kmip/spec/v1.2/kmip-spec-v1.2.pdf

Technical Committee:

OASIS Key Management Interoperability Protocol (KMIP) TC

Chairs:

Subhash Sankuratripati (Subhash.Sankuratripati@netapp.com), NetApp
Saikat Saha (saikat.saha@oracle.com), Oracle

Editors:

Kiran Thota (kthota@vmware.com), VMware, Inc.
Kelley Burgin (kwburgi@tycho.ncsc.mil), National Security Agency

Related work:

This specification replaces or supersedes:

- *Key Management Interoperability Protocol Specification Version 1.0*. Edited by Robert Haas and Indra Fitzgerald. 01 October 2010. OASIS Standard. http://docs.oasis-open.org/kmip/spec/v1.0/os/kmip-spec-1.0-os.html.
- *Key Management Interoperability Protocol Specification Version 1.1.* Edited by Robert Haas and Indra Fitzgerald. 24 January 2013. OASIS Standard. http://docs.oasis-open.org/kmip/spec/v1.1/os/kmip-spec-v1.1-os.html.

This specification is related to:

- *Key Management Interoperability Protocol Profiles Version 1.2*. Edited by Tim Hudson and Robert Lockhart. Latest version: http://docs.oasis-open.org/kmip/profiles/v1.2/kmip-profiles-v1.2.html.
- *Key Management Interoperability Protocol Test Cases Version 1.2*. Edited by Tim Hudson and Faisal Faruqui. Latest version: http://docs.oasis-open.org/kmip/testcases/v1.2/kmip-testcases-v1.2.html.

- *Key Management Interoperability Protocol Usage Guide Version 1.2*. Edited by Indra Fitzgerald and Judith Furlong. Latest version: http://docs.oasis-open.org/kmip/ug/v1.2/kmip-ug-v1.2.html.
- *KMIP Tape Library Profile Version 1.0*. Edited by Tim Hudson, Stan Feather, and Rod Wideman. Latest version: http://docs.oasis-open.org/kmip/kmip-tape-lib-profile/v1.0/kmip-tape-lib-profile-v1.0.html.
- *KMIP Symmetric Key Lifecycle Profile Version 1.0*. Edited by Tim Hudson and Robert Lockhart. Latest version: http://docs.oasis-open.org/kmip/kmip-sym-key-profile/v1.0/kmip-sym-key-profile-v1.0.html.
- *KMIP Symmetric Key Foundry for FIPS 140-2 Profile Version 1.0*. Edited by Tim Hudson and Robert Lockhart. Latest version: http://docs.oasis-open.org/kmip/kmip-sym-foundry-profile/v1.0/kmip-sym-foundry-profile-v1.0.html.
- *KMIP Suite B Profile Version 1.0*. Edited by Kelley Burgin and Tim Hudson. Latest version: http://docs.oasis-open.org/kmip/kmip-suite-b-profile/v1.0/kmip-suite-b-profile-v1.0.html.
- *KMIP Storage Array with Self-Encrypting Drives Profile Version 1.0*. Edited by Tim Hudson and Mahadev Karadigudda. Latest version: http://docs.oasis-open.org/kmip/kmip-sa-sed-profile/v1.0/kmip-sa-sed-profile-v1.0.html.
- *KMIP Opaque Managed Object Store Profile Version 1.0*. Edited by Tim Hudson and Robert Lockhart. Latest version: http://docs.oasis-open.org/kmip/kmip-opaque-obj-profile/v1.0/kmip-opaque-obj-profile-v1.0.html.
- KMIP Cryptographic Services Profile Version 1.0. Edited by Tim Hudson. Latest version: http://docs.oasis-open.org/kmip/kmip-cs-profile/v1.0/kmip-cs-profile-v1.0.html.
- KMIP Asymmetric Key Lifecycle Profile Version 1.0. Edited by Tim Hudson and Robert Lockhart. Latest version: http://docs.oasis-open.org/kmip/kmip-asym-key-profile/v1.0/kmip-asym-key-profile-v1.0.html.

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 OASIS Key Management Interoperability Protocol (KMIP) TC on the above date. The level of approval is also listed above. Check the “Latest 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 https://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 (https://www.oasis-open.org/committees/kmip/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[kmip-spec-v1.2]

Key Management Interoperability Protocol Specification Version 1.2. Edited by Kiran Thota and Kelley Burgin. 19 June 2014. OASIS Committee Specification Draft 02 / Public Review Draft 02. http://docs.oasis-open.org/kmip/spec/v1.2/csprd02/kmip-spec-v1.2-csprd02.html. Latest version: http://docs.oasis-open.org/kmip/spec/v1.2/kmip-spec-v1.2.html.

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., SCVP or XKMS), Certify (e.g., CMP **[RFC4210]**, CMC **[RFC5272][RFC6402]**, SCEP) and Re-certify (e.g., CMP **[RFC4210]**, CMC **[RFC5272][RFC6402]**, 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.

This specification is complemented by several other documents. The KMIP Usage Guide**[KMIP-UG]** provides illustrative information on using the protocol. The KMIP Profiles Specification **[KMIP-Prof]** provides a selected set of base level conformance profiles and authentication suites; additional KMIP Profiles define specific sets of KMIP functionality for conformance purposes. The KMIP Test Specification **[KMIP-TC]** provides samples of protocol messages corresponding to a set of defined test cases. The KMIP Use Cases document **[KMIP-UC]** provides user stories that define the use of and context for functionality defined in KMIP.

This specification defines the KMIP protocol version major 1 and minor 2 (see 6.1).

1.1 Terminology

The key words “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119][RFC2119]**.

For acronyms used in this document, see Appendix E. 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 a security function.
Authorization	Access privileges that are granted to an entity; conveying an “official” sanction to perform a security function or activity.
Certificate length	The length (in bytes) of an X.509 public key certificate.
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, and 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 hashing algorithm 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.
Digital Signature Algorithm	A cryptographic algorithm used for digital signature.
Encryption	The process of changing plaintext into ciphertext using a cryptographic algorithm and key.
Hashing algorithm (or hash algorithm, hash function)	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: 1) Either a shared secret from a key agreement computation or a pre-shared cryptographic key, and 2) Other information.
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.
Message Authentication Code (MAC)	A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data.
PGP Key	A RFC 4880-compliant container of cryptographic keys and associated metadata. Usually text-based (in PGP-parlance, ASCII-armored).
Private key	A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and is not made public. 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 can 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. 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	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

algorithm	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 Key	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 necessary to construct the original key, then knowledge of any *k*-1 key components provides no information about the original key other than, possibly, 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 inverse (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.

Table 1: Terminology

1.2 Normative References

[ECC-Brainpool] *ECC Brainpool Standard Curves and Curve Generation v. 1.0.19.10.2005*, http://www.ecc-brainpool.org/download/Domain-parameters.pdf.

[FIPS180-4] *Secure Hash Standard (SHS)*, FIPS PUB 186-4, March 2012, http://csrc.nist.gov/publications/fips/fips18-4/fips-180-4.pdf.

[FIPS186-4] *Digital Signature Standard (DSS)*, FIPS PUB 186-4, July 2013, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[FIPS197] *Advanced Encryption Standard*, FIPS PUB 197, November 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, 2012.

[ISO9797-1] ISO/IEC, *Information technology -- Security techniques -- Message Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher*, ISO/IEC 9797-1, 2011.

[KMIP-Prof] *Key Management Interoperability Protocol Profiles Version 1.2.* Edited by Tim Hudson and Robert Lockhart. Latest version: http://docs.oasis-open.org/kmip/profiles/v1.2/kmip-profiles-v1.2.html.

[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#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, April 1992, http://www.ietf.org/rfc/rfc1320.txt.

[RFC1321] R. Rivest, *The MD5 Message-Digest Algorithm*, IETF RFC 1321, April 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, February 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, Feb 1993, http://www.ietf.org/rfc/rfc1424.txt.

[RFC2104] H. Krawczyk, M. Bellare, R. Canetti, *HMAC: Keyed-Hashing for Message Authentication*, IETF RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.txt.

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

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

[RFC2986] M. Nystrom and B. Kaliski, *PKCS #10: Certification Request Syntax Specification Version 1.7*, IETF RFC2986, November 2000, http://www.rfc-editor.org/rfc/rfc2986.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, November 2003, http://www.ietf.org/rfc/rfc3629.txt.

[RFC3686] R. Housley, *Using Advanced Encryption Standard (AES) Counter Mode with IPsec Encapsulating Security Payload (ESP), IETF RFC 3686,* January 2004, http://www.ietf.org/rfc/rfc3686.txt.

[RFC4210] C. Adams, S. Farrell, T. Kause and T. Mononen, *Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)*, IETF RFC 4210, 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, September 2005, http://www.ietf.org/rfc/rfc4211.txt.

[RFC4880] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, *OpenPGP Message Format*, IETF RFC 4880, November 2007, http://www.ietf.org/rfc/rfc4880.txt.

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

[RFC5208] B. Kaliski, *Public Key Cryptographic Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2*, IETF RFC5208, May 2008, http://www.rfc-editor.org/rfc/rfc5208.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.

[RFC5639] M. Lochter, J. Merkle, *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*, IETF RFC 5639, March 2010, http://www.ietf.org/rfc/rfc5639.txt.

[RFC6402] J. Schaad, *Certificate Management over CMS (CMC) Updates*, IETF RFC6402, November 2011, http://www.rfc-editor.org/rfc/rfc6402.txt.

[RFC6818] P. Yee, *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, IETF RFC6818, January 2013, http://www.rfc-editor.org/rfc/rfc6818.txt.

[SEC2] SEC 2: Recommended Elliptic Curve Domain Parameters, http://www.secg.org/collateral/sec2_final.pdf.

[SP800-38A] M. Dworkin, *Recommendation for Block Cipher Modes of Operation – Methods and Techniques*, NIST Special Publication 800-38A, December 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, January 2010, http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf.

[SP800-56A] E. Barker, L. Chen, A. Roginsky and M. Smid, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, NIST Special Publication 800-56A Revision 2, May 2013, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf.

[SP800-57-1] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendations for Key Management - Part 1: General (Revision 3),* NIST Special Publication 800-57 Part 1 Revision 3, July 2012, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

[SP800-108] L. Chen, *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*, NIST Special Publication 800-108, Oct 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, November 2008, http://www.itu.int/rec/recommendation.asp?lang=en&parent=T-REC-X.509-200811-1.

[X9.24-1] ANSI, *X9.24 - Retail Financial Services Symmetric Key Management - Part 1: Using Symmetric Techniques*, 2009.

[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.

[X9.62] 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*, 2011.

[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*, 2010.

1.3 Non-Normative References

[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.

[KMIP-UG] *Key Management Interoperability Protocol Usage Guide Version 1.2.* Edited by Indra Fitzgerald and Judith Furlong. Latest version: http://docs.oasis-open.org/kmip/ug/v1.2/kmip-ug-v1.2.html.

[KMIP-TC] *Key Management Interoperability Protocol Test Cases Version 1.2.* Edited by Tim Hudson and Faisal Faruqui. Latest version: http://docs.oasis-open.org/kmip/testcases/v1.2/kmip-testcases-v1.2.html.

[KMIP-UC] *Key Management Interoperability Protocol Use Cases Version 1.2 Working Draft 10*, June 20, 2013. https://www.oasis-open.org/committees/download.php/49644/kmip-usecases-v1.2-wd10.doc.

[RFC6151] S. Turner and L. Chen, *Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms*, IETF RFC6151, March 2011, http://www.rfc-editor.org/rfc/rfc6151.txt.

[w1979] A. Shamir, *How to share a secret*, Communications of the ACM, vol. 22, no. 11, pp. 612-613, November 1979.

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 (see Section 9.1.1.4):

- 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

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 2) 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. 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. Single-instance Attributes (attributes which an object MAY only have at most one instance thereof) SHALL have an Attribute Index of 0. The *Attribute Value* is either a primitive data type or structured object, depending on the attribute.

When an Attribute structure is used to specify or return a particular instance of an Attribute and the Attribute Index is not specified it SHALL be assumed to be 0.

Object	Encoding	REQUIRED
Attribute	Structure	
Attribute Name	Text String	Yes
Attribute Index	Integer	No
Attribute Value	Varies, depending on attribute. See Section 3	Yes, except for the Notify operation (see Section 5.1)

Table 2: Attribute Object Structure

2.1.2 Credential

A *Credential* is a structure (see Table 3) 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]**.

Object	Encoding	REQUIRED
Credential	Structure	
Credential Type	Enumeration, see 9.1.3.2.1	Yes
Credential Value	Varies based on Credential Type.	Yes

Table 3: Credential Object Structure

If the Credential Type in the Credential is *Username and Password*, then Credential Value is a structure as shown in Table 4. The Username field identifies the client, and the Password field is a secret that authenticates the client.

Object	Encoding	REQUIRED
Credential Value	Structure	
Username	Text String	Yes
Password	Text String	No

Table 4: Credential Value Structure for the Username and Password Credential

If the Credential Type in the Credential is *Device*, then Credential Value is a structure as shown in Table 5. One or a combination of the *Device Serial Number*, *Network Identifier*, *Machine Identifier*, and *Media Identifier* SHALL be unique. Server implementations MAY enforce policies on uniqueness for individual fields. A shared secret or password MAY also be used to authenticate the client. The client SHALL provide at least one field.

Object	Encoding	REQUIRED
Credential Value	Structure	
Device Serial Number	Text String	No
Password	Text String	No
Device Identifier	Text String	No
Network Identifier	Text String	No
Machine Identifier	Text String	No
Media Identifier	Text String	No

Table 5: Credential Value Structure for the Device Credential

If the Credential Type in the Credential is *Attestation*, then Credential Value is a structure as shown in Table 6. The *Nonce Value* is obtained from the key management server in a Nonce Object. The Attestation Credential Object can contain a measurement from the client or an assertion from a third party if the server is not capable or willing to verify the attestation data from the client. Neither type of attestation data (*Attestation Measurement* or *Attestation Assertion*) is necessary to allow the server to accept either. However, the client SHALL provide attestation data in either the *Attestation Measurement* or *Attestation Assertion* fields.

Object	Encoding	REQUIRED
Credential Value	Structure	
Nonce	Structure, see 2.1.14	Yes
Attestation Type	Enumeration, see 9.1.3.2.36	Yes
Attestation Measurement	Byte String	No
Attestation Assertion	Byte String	No

Table 6: Credential Value Structure for the Attestation Credential

2.1.3 Key Block

A *Key Block* object is a structure (see Table 7) 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*:

- *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.

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 from 112 to 192 bits (depending upon key length and the presence of parity bits).
- AES keys are 128, 192 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	No
Cryptographic Algorithm	Enumeration, see 9.1.3.2.13	Yes. MAY be omitted only if this information is available from the Key Value. Does not apply to Secret Data (see Section 2.2.7) or Opaque Objects (see Section 2.2.8). 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 (see Section 2.2.7) or Opaque Objects (see Section 2.2.8). 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.

Table 7: 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 8):

- 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 is obtained via the Get Attributes operation, only by the fact that it is encapsulated with (and possibly wrapped with) the key material itself.
- The Key Value Byte String is either the wrapped TTLV-encoded (see Section 9.1) Key Value structure, or the wrapped un-encoded value of the Byte String Key Material field.

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

Table 8: 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 9). It is only used inside a Key Block.

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.
- An *Encoding Option*, specifying the encoding of the Key Material within the Key Value structure of the Key Block that has been wrapped. If No Encoding is specified, then the Key Value structure SHALL NOT contain any attributes.

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). Either the Encryption Key Information or the MAC/Signature Key Information (or both) in the Key Wrapping Data structure 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.*

The following encoding options are currently defined:

- *No Encoding* (i.e., the wrapped un-encoded value of the Byte String Key Material field in the Key Value structure).
- *TTLV Encoding* (i.e., the wrapped TTLV-encoded Key Value structure).

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
Encoding Option	Enumeration, see 9.1.3.2.32	No. Specifies the encoding of the Key Value Byte String. If not present, the wrapped Key Value structure SHALL be TTLV encoded.

Table 9: Key Wrapping Data Object Structure

The structures of the Encryption Key Information (see Table 10) and the MAC/Signature Key Information (see Table 11) are as follows:

Object	Encoding	REQUIRED
Encryption Key Information	Structure	
Unique Identifier	Text string, see 3.1	Yes
Cryptographic Parameters	Structure, see 3.6	No

Table 10: Encryption Key Information Object Structure

Object	Encoding	REQUIRED
MAC/Signature Key Information	Structure	
Unique Identifier	Text string, see 3.1	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	Structure, see 3.6	No

Parameters		

Table 11: MAC/Signature Key Information Object Structure

2.1.6 Key Wrapping Specification

This is a separate structure (see Table 12) 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 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.

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.
- An Encoding Option, specifying the encoding of the Key Value before wrapping. If No Encoding is specified, then the Key Value SHALL NOT contain any attributes

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
Encoding Option	Enumeration, see 9.1.3.2.32	No. If Encoding Option is not present, the wrapped Key Value SHALL be TTLV encoded.

Table 12: 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. The mapping to the parameters specified in other standards is shown in Table 13.

Object	Description	Mapping
P	For DSA and DH, the (large) prime field order.	p in **[FIPS186-4]**, **[X9.42]**,

	For RSA, a prime factor of the modulus.	**[SP800-56A]** p in **[PKCS#1]**, **[FIPS186-4]**
Q	For DSA and DH, the (small) prime multiplicative subgroup order. For RSA, a prime factor of the modulus.	q in **[FIPS186-4]**, **[X9.42]**, **[SP800-56A]** q in **[PKCS#1]**, **[FIPS186-4]**
G	The generator of the subgroup of order Q.	g in **[FIPS186-4]**, **[X9.42]**, **[SP800-56A]**
X	DSA or DH private key.	x in **[FIPS186-4]** x, $x_{u,}$ x_v in **[X9.42]**, **[SP800-56A]** for static private keys r, $r_{u,}$ r_v in **[X9.42]**, **[SP800-56A]** for ephemeral private keys
Y	DSA or DH public key.	y in **[FIPS186-4]** y, $y_{u,}$ y_v in **[X9.42]**, **[SP800-56A]** for static public keys t, $t_{u,}$ t_v in **[X9.42]**, **[SP800-56A]** for ephemeral public keys
J	DH cofactor integer, where P = JQ + 1.	j in **[X9.42]**
Modulus	RSA modulus PQ, where P and Q are distinct primes.	n in **[PKCS#1]**, **[FIPS186-4]**
Private Exponent	RSA private exponent.	d in **[PKCS#1]**, **[FIPS186-4]**
Public Exponent	RSA public exponent.	e in **[PKCS#1]**, **[FIPS186-4]**
Prime Exponent P	RSA private exponent for the prime factor P in the CRT format, i.e., Private Exponent (mod (P-1)).	dP in **[PKCS#1]**, **[FIPS186-4]**
Prime Exponent Q	RSA private exponent for the prime factor Q in the CRT format, i.e., Private Exponent (mod (Q-1)).	dQ in **[PKCS#1]**, **[FIPS186-4]**
CRT Coefficient	The (first) CRT coefficient, i.e., Q^{-1} mod P.	qInv in **[PKCS#1]**, **[FIPS186-4]**
Recommended Curve	NIST Recommended Curves (e.g., P-192).	See Appendix D of **[FIPS186-4]**
D	Elliptic curve private key.	d; $d_{e,U}$,$d_{e,V}$ (ephemeral private keys); $d_{s,U}$,$d_{s,V}$ (static private keys) in **[X9.62]**, **[FIPS186-4]**
Q String	Elliptic curve public key.	Q; $Q_{e,U}$,$Q_{e,V}$ (ephemeral public keys); $Q_{s,U}$,$Q_{s,V}$ (static public keys) in **[X9.62]**, **[FIPS186-4]**

Table 13: Parameter mapping.

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 14.

Object	Encoding	REQUIRED
Key Material	Structure	
Key	Byte String	Yes

Table 14: 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 15.

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

Table 15: Key Material Object Structure for Transparent DSA Private Keys

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 16.

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

Table 16: Key Material Object Structure for Transparent DSA Public Keys

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 17.

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 17: Key Material Object Structure for Transparent RSA Private Keys

One of the following SHALL be present (refer to **[PKCS#1]**):

- Private Exponent,
- P and Q (the first two prime factors of Modulus), or
- 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 18.

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

Table 18: 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 19.

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

Table 19: Key Material Object Structure for Transparent DH Private Keys

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 20.

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

Table 20: Key Material Object Structure for Transparent DH Public Keys

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 21.

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

Table 21: Key Material Object Structure for Transparent ECDSA Private Keys

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 22.

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

Table 22: Key Material Object Structure for Transparent ECDSA Public Keys

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 23.

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

Table 23: 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 24.

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

Table 24: Key Material Object Structure for Transparent ECDH Public Keys

2.1.7.12 Transparent ECMQV Private Key

If the Key Format Type in the Key Block is *Transparent ECMQV Private Key*, then Key Material is a structure as shown in Table 25.

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

Table 25: 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 26.

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

Table 26: 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

Table 27: Template-Attribute Object Structure

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

2.1.9 Extension Information

An *Extension Information* object is a structure (see Table 28) describing Objects with Item Tag values in the Extensions range. The Extension Name is a Text String that is used to name the Object (first column of Table 251). The Extension Tag is the Item Tag Value of the Object (see Table 251). The Extension Type is the Item Type Value of the Object (see Table 249).

Object	Encoding	REQUIRED
Extension Information	Structure	
Extension Name	Text String	Yes
Extension Tag	Integer	No
Extension Type	Integer	No

Table 28: Extension Information Structure

2.1.10 Data

The *Data* object is used in requests and responses in cryptographic operations that pass data between the client and the server.

Object	Encoding
Data	Byte String

Table 29: Data Structure

2.1.11 Data Length

The *Data Length* is used in requests in cryptographic operations to indicate the amount of data expected in a response.

Object	Encoding
Data Length	Integer

Table 30: Data Length Structure

2.1.12 Signature Data

The *Signature Data* is used in requests and responses in cryptographic operations that pass signature data between the client and the server.

Object	Encoding
Signature Data	Byte String

Table 31: Signature Data Structure

2.1.13 MAC Data

The *MAC Data* is used in requests and responses in cryptographic operations that pass MAC data between the client and the server.

Object	Encoding
MAC Data	Byte String

Table 32: MAC Data Structure

2.1.14 Nonce

A *Nonce* object is a structure (see Table 33) used by the server to send a random value to the client. The Nonce Identifier is assigned by the server and used to identify the Nonce object. The Nonce Value consists of the random data created by the server.

Object	Encoding	REQUIRED
Nonce	Structure	
Nonce ID	Byte String	Yes
Nonce Value	Byte String	Yes

Table 33: Nonce Structure

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).

2.2.1 Certificate

A Managed Cryptographic Object that is a digital certificate. It is a DER-encoded X.509 public key certificate. The PGP certificate type is deprecated as of version 1.2 of this specification and MAY be removed from subsequent versions of the specification. The PGP Key object (see section 2.2.9) SHOULD be used instead.

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

Table 34: 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

Table 35: 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

Table 36: 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

Table 37: 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 9.1.3.2.8	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

Table 38: 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 **[w1979]**.

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):

  $GF(2^{16}) \approx GF(2^8)\,[y]/(y^2+y+m)$, 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 **[FIPS197]**, Sections 3 and 4. The field GF(2^8) is as described in **[FIPS197]**,

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

  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 **[FIPS197]** 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 **[FIPS197]**).

  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. A Template is used to specify the attributes of a new Managed Cryptographic Object in various operations. Attributes associated with a Managed Object MAY also be specified in the Template-Attribute structures in the operations in Section 4.

Attributes specified in a Template apply to any object created that reference the Template by name using the Name object in any of the Template-Attribute structures in Section 2.1.8.

The name of a Template (as it is for any Managed Object) is specified as an Attribute in the Template-Attribute structure in the Register operation where the Attribute Name is "Name" and the Attribute Value is the name of the Template Managed Object.

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

Table 39: 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 Secret Data Type. The Key Value MAY be wrapped.

Object	Encoding	REQUIRED
Secret Data	Structure	
Secret Data Type	Enumeration, see 9.1.3.2.9	Yes
Key Block	Structure, see 2.1.3	Yes

Table 40: 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 9.1.3.2.10	Yes
Opaque Data Value	Byte String	Yes

Table 41: Opaque Object Structure

2.2.9 PGP Key

A Managed Cryptographic Object that is a text-based representation of a PGP key. The Key Block field, indicated below, will contain the ASCII-armored export of a PGP key in the format as specified in RFC 4880. It MAY contain only a public key block, or both a public and private key block. Two different versions of PGP keys, version 3 and version 4, MAY be stored in this Managed Cryptographic Object.

KMIP implementers SHOULD treat the Key Block field as an opaque blob. PGP-aware KMIP clients SHOULD take on the responsibility of decomposing the Key Block into other Managed Cryptographic Objects (Public Keys, Private Keys, etc.).

Object	Encoding	REQUIRED
PGP Key	Structure	
PGP Key Version	Integer	Yes
Key Block	Structure, see 2.1.3	Yes

Table 42: PGP Key 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*. All instances of an attribute SHOULD have a different value. Similarly, attributes which an object SHALL only have at most one instance of are referred to as *single-instance attributes*. 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.

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).

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.

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 43 below explains the meaning of each characteristic that MAY appear in those tables. The server policy MAY further restrict these attribute characteristics.

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 43: 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 this identifier SHOULD 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 44: Unique Identifier 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, Re-key Key Pair
Applies to Object Types	All Objects

Table 45: Unique Identifier Attribute Rules

3.2 Name

The *Name* attribute is a structure (see Table 46) used to identify and locate an 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.

Object	Encoding	REQUIRED
Name	Structure	
Name Value	Text String	Yes
Name Type	Enumeration, see 9.1.3.2.11	Yes

Table 46: 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-key Key Pair, Re-certify
Applies to Object Types	All Objects

Table 47: 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.

Object	Encoding
Object Type	Enumeration, see 9.1.3.2.12

Table 48: 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, Re-key Key Pair
Applies to Object Types	All Objects

Table 49: Object Type Attribute Rules

3.4 Cryptographic Algorithm

The *Cryptographic Algorithm* of an object. The Cryptographic Algorithm of a Certificate object identifies the algorithm for the public key contained within the Certificate. The digital signature algorithm used to sign the Certificate is identified in the Digital Signature Algorithm attribute defined in Section 3.16. 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 9.1.3.2.13

Table 50: 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	Certify, Create, Create Key Pair, Re-certify, Register, Derive Key, Re-key, Re-key Key Pair
Applies to Object Types	Keys, Certificates, Templates

Table 51: Cryptographic Algorithm Attribute Rules

3.5 Cryptographic Length

For keys, *Cryptographic Length* is the length in bits of the clear-text cryptographic key material of the Managed Cryptographic Object. For certificates, *Cryptographic Length* is the length in bits of the public key contained within the Certificate. 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 52: 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	Certify, Create, Create Key Pair, Re-certify, Register, Derive Key, Re-key, Re-key Key Pair
Applies to Object Types	Keys, Certificates, Templates

Table 53: Cryptographic Length Attribute Rules

3.6 Cryptographic Parameters

The *Cryptographic Parameters* attribute is a structure (see Table 54) 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. The Cryptographic Parameters attribute of a Certificate object identifies the cryptographic parameters of the public key contained within the Certificate.

The Cryptographic Algorithm is also used to specify the parameters for cryptographic operations. For operations involving digital signatures, either the Digital Signature Algorithm can be specified or the Cryptographic Algorithm and Hashing Algorithm combination can be specified.

Random IV can be used to request that the KMIP server generate an appropriate IV for a cryptographic operation that uses an IV. The generated Random IV is returned in the response to the cryptographic operation.

IV Length is the length of the Initialization Vector in bits. This parameter SHALL be provided when the specified Block Cipher Mode supports variable IV lengths such as CTR or GCM.

Tag Length is the length of the authenticator tag in bytes. This parameter SHALL be provided when the Block Cipher Mode is GCM.

The IV used with counter modes of operation (e.g., CTR and GCM) cannot repeat for a given cryptographic key. To prevent an IV/key reuse, the IV is often constructed of three parts: a fixed field, an invocation field, and a counter as described in **[SP800-38A]** and **[SP800-38D]**. The Fixed Field Length is the length of the fixed field portion of the IV in bits. The Invocation Field Length is the length of the invocation field portion of the IV in bits. The Counter Length is the length of the counter portion of the IV in bits.

Initial Counter Value is the starting counter value for CTR mode (for **[RFC3686]** it is 1).

Object	Encoding	REQUIRED
Cryptographic Parameters	Structure	
Block Cipher Mode	Enumeration, see 9.1.3.2.14	No
Padding Method	Enumeration, see 9.1.3.2.15	No
Hashing Algorithm	Enumeration, see 9.1.3.2.16	No
Key Role Type	Enumeration, see 9.1.3.2.17	No
Digital Signature Algorithm	Enumeration, see 9.1.3.2.7	No
Cryptographic Algorithm	Enumeration, see 9.1.3.2.13	No
Random IV	Boolean	No
IV Length	Integer	No unless Block Cipher Mode supports variable IV lengths
Tag Length	Integer	No unless Block Cipher Mode is GCM
Fixed Field Length	Integer	No
Invocation Field Length	Integer	No
Counter Length	Integer	No
Initial Counter Value	Integer	No

612 *Table 54: 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-key Key Pair, Re-certify
Applies to Object Types	Keys, Certificates, Templates

613 *Table 55: Cryptographic Parameters Attribute Rules*

614 Key Role Type definitions match those defined in ANSI X9 TR-31 **[X9 TR-31]** and are defined in Table
615 56:

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 56: Key Role Types

Accredited Standards Committee X9, Inc. - Financial Industry Standards (www.x9.org) contributed to Table 56. Key role names and descriptions are derived from material in the Accredited Standards Committee X9, Inc.'s Technical Report "TR-31 2010 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.

3.7 Cryptographic Domain Parameters

The *Cryptographic Domain Parameters* attribute is a structure (see Table 57) 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.

The domain parameter Qlength correponds to the bit length of parameter Q (refer to **[SEC2]** and **[SP800-56A]**). Qlength applies to algorithms such as DSA and DH. The bit length of parameter P (refer to **[SEC2]** and **[SP800-56A]**) is specified separately by setting the Cryptographic Length attribute.

Recommended Curve is applicable to elliptic curve algorithms such as ECDSA, ECDH, and ECMQV.

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

Table 57: 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, Re-key Key Pair
Applies to Object Types	Asymmetric Keys, Templates

Table 58: Cryptographic Domain Parameters Attribute Rules

3.8 Certificate Type

The *Certificate Type* attribute is a type of certificate (e.g., X.509). The PGP certificate type is deprecated as of version 1.2 of this specification and MAY be removed from subsequent versions of the specification.

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	

Table 59: 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 60: Certificate Type Attribute Rules

3.9 Certificate Length

The *Certificate Length* attribute is the length in bytes of the Certificate object. The *Certificate Length* 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
Certificate Length	Integer

Table 61: Certificate 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	Register, Certify, Re-certify
Applies to Object Types	Certificates

Table 62: Certificate Length Attribute Rules

3.10 X.509 Certificate Identifier

The *X.509 Certificate Identifier* attribute is a structure (see Table 63) used to provide the identification of an X.509 public key certificate. The X.509 Certificate Identifier contains the Issuer Distinguished Name (i.e., from the Issuer field of the X.509 certificate) and the Certificate Serial Number (i.e., from the Serial Number field of the X.509 certificate). The X.509 Certificate Identifier SHALL be set by the server when the X.509 certificate is created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

Object	Encoding	REQUIRED
X.509 Certificate Identifier	Structure	
Issuer Distinguished Name	Byte String	Yes
Certificate Serial Number	Byte String	Yes

Table 63: X.509 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	X.509 Certificates

Table 64: X.509 Certificate Identifier Attribute Rules

3.11 X.509 Certificate Subject

The *X.509 Certificate Subject* attribute is a structure (see Table 65) used to identify the subject of a X.509 certificate. The X.509 Certificate Subject contains the Subject Distinguished Name (i.e., from the Subject field of the X.509 certificate). It MAY include one or more alternative names (e.g., email address, IP address, DNS name) for the subject of the X.509 certificate (i.e., from the Subject Alternative Name extension within the X.509 certificate). The X.509 Certificate Subject SHALL be set by the server based on the information it extracts from the X.509 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.

If the Subject Alternative Name extension is included in the X.509 certificate and is marked critical within the X.509 certificate itself, then an X.509 certificate MAY be issued with the subject field left blank. Therefore an empty string is an acceptable value for the Subject Distinguished Name.

Object	Encoding	REQUIRED
X.509 Certificate Subject	Structure	
Subject Distinguished Name	Byte String	Yes, but MAY be the empty string
Subject Alternative Name	Byte String	Yes, if the Subject Distinguished Name is an empty string. MAY be repeated

Table 65: X.509 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	X.509 Certificates

Table 66: X.509 Certificate Subject Attribute Rules

3.12 X.509 Certificate Issuer

The *X.509 Certificate Issuer* attribute is a structure (see Table 71) used to identify the issuer of a X.509 certificate, containing the Issuer Distinguished Name (i.e., from the Issuer field of the X.509 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 X.509 certificate). The server SHALL set these values based on the information it extracts from a X.509 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.

Object	Encoding	REQUIRED
X.509 Certificate Issuer	Structure	
Issuer Distinguished Name	Byte String	Yes
Issuer Alternative Name	Byte String	No, MAY be repeated

Table 67: X.509 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	X.509 Certificates

Table 68: X.509 Certificate Issuer Attribute Rules

3.13 Certificate Identifier

This attribute is deprecated as of version 1.1 of this specification and MAY be removed from subsequent versions of this specification. The X.509 Certificate Identifier attribute (see Section 3.10) SHOULD be used instead.

The *Certificate Identifier* attribute is a structure (see Table 69) used to provide the identification of a certificate. For X.509 certificates, it contains 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). For PGP certificates, the Issuer contains the OpenPGP Key ID of the key issuing the signature (the signature that represents the certificate). The Certificate Identifier 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 69: 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 70: Certificate Identifier Attribute Rules

3.14 Certificate Subject

This attribute is deprecated as of version 1.1 of this specification and MAY be removed from subsequent versions of this specification. The X.509 Certificate Subject attribute (see Section 3.11) SHOULD be used instead.

The *Certificate Subject* attribute is a structure (see Table 71) used to identify the subject of a certificate. For X.509 certificates, it contains 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). For PGP certificates, the Certificate Subject Distinguished Name contains the content of the first User ID packet in the PGP certificate (that is, the first User ID packet after the Public-Key packet in the transferable public key that forms the PGP 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.

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 71: 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 72: Certificate Subject Attribute Rules

3.15 Certificate Issuer

This attribute is deprecated as of version 1.1 of this specification and MAY be removed from subsequent versions of this specification. The X.509 Certificate Issuer attribute (see Section 3.12) SHOULD be used instead.

The *Certificate Issuer* attribute is a structure (see Table 74) 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.

Object	Encoding	REQUIRED
Certificate Issuer	Structure	
Certificate Issuer Distinguished Name	Text String	Yes

Certificate Issuer Alternative Name	Text String	No, MAY be repeated

Table 73: 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 74: Certificate Issuer Attribute Rules

3.16 Digital Signature Algorithm

The *Digital Signature Algorithm* attribute identifies the digital signature algorithm associated with a digitally signed object (e.g., Certificate). 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
Digital Signature Algorithm	Enumeration, see 9.1.3.2.7

Table 75: Digital Signature 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	Yes for PGP keys. No for X.509 certificates.
When implicitly set	Certify, Re-certify, Register
Applies to Object Types	Certificates, PGP keys

Table 76: Digital Signature Algorithm Attribute Rules

3.17 Digest

The *Digest* attribute is a structure (see Table 77) 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). If the Key Material is a Byte String, then the Digest Value SHALL be calculated on this Byte String. If the Key Material is a structure, then the Digest Value SHALL be calculated on the TTLV-encoded (see Section 9.1) Key Material structure. The Key Format Type field in the Digest attribute indicates the format of the Managed Object from which the Digest Value was calculated. Multiple digests MAY be calculated using different algorithms listed in Section 9.1.3.2.16 and/or key format types listed in Section 9.1.3.2.3. If this attribute exists, then it SHALL have a mandatory attribute instance computed with the SHA-256 hashing algorithm. For objects registered by a client, the server SHALL compute the digest of the mandatory attribute instance using the Key Format Type of the registered object. In all other cases, the server MAY use any Key Format Type when computing the digest of the mandatory attribute instance, provided it is able to serve the object to clients in that same format. The digest(s) are static and SHALL be set by the server when the object is created or registered, provided that the server has access to the Key Material or the Digest Value (possibly obtained via out-of-band mechanisms).

Object	Encoding	REQUIRED
Digest	Structure	
Hashing Algorithm	Enumeration, see 9.1.3.2.16	Yes
Digest Value	Byte String	Yes, if the server has access to the Digest Value or the Key Material (for keys and secret data), the Certificate Value (for certificates) or the Opaque Data Value (for opaque objects).
Key Format Type	Enumeration, see 9.1.3.2.3	Yes, if the Managed Object is a key or secret data object.

Table 77: Digest Attribute Structure

SHALL always have a value	Yes, if the server has access to the Digest Value or the Key Material (for keys and secret data), the Certificate Value (for certificates) or the Opaque Data Value (for opaque objects).
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, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects, Opaque Objects

Table 78: Digest Attribute Rules

3.18 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.

Object	Encoding
Operation Policy Name	Text String

Table 79: 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, Re-key Key Pair
Applies to Object Types	All Objects

Table 80: Operation Policy Name Attribute Rules

3.18.1 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
- Re-certify
- Validate
- Query
- Cancel
- Poll

3.18.2 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 owner SHALL be as defined in **[KMIP-Prof]**.

3.18.2.1 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 owner only
Re-key Key Pair	Allowed to owner only
Derive Key	Allowed to owner only
Locate	Allowed to owner only
Check	Allowed to owner only
Get	Allowed to owner only
Get Attributes	Allowed to owner only
Get Attribute List	Allowed to owner only
Add Attribute	Allowed to owner only
Modify Attribute	Allowed to owner only
Delete Attribute	Allowed to owner only
Obtain Lease	Allowed to owner only
Get Usage Allocation	Allowed to owner only
Activate	Allowed to owner only
Revoke	Allowed to owner only
Destroy	Allowed to owner only
Archive	Allowed to owner only
Recover	Allowed to owner only

Table 81: Default Operation Policy for Secret Objects

3.18.2.2 Default Operation Policy for Certificates and Public Key Objects

This policy applies to Certificates and Public Keys.

Default Operation Policy for Certificates and Public Key Objects	
Operation	**Policy**
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 owner only
Modify Attribute	Allowed to owner only
Delete Attribute	Allowed to owner only
Obtain Lease	Allowed to all

Activate	Allowed to owner only
Revoke	Allowed to owner only
Destroy	Allowed to owner only
Archive	Allowed to owner only
Recover	Allowed to owner only

Table 82: Default Operation Policy for Certificates and Public Key Objects

3.18.2.3 Default Operation Policy for Template Objects

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.

Default Operation Policy for Private Template Objects	
Operation	**Policy**
Locate	Allowed to owner only
Get	Allowed to owner only
Get Attributes	Allowed to owner only
Get Attribute List	Allowed to owner only
Add Attribute	Allowed to owner only
Modify Attribute	Allowed to owner only
Delete Attribute	Allowed to owner only
Destroy	Allowed to owner only
Any operation referencing the Template using a Template-Attribute	Allowed to owner only

Table 83: 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

Any operation referencing the Template using a Template-Attribute	Allowed to all

Table 84: Default Operation Policy for Public Template Objects

3.19 Cryptographic Usage Mask

The *Cryptographic Usage Mask* attribute 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 or Verify
contentCommitment	Content Commitment (Non Repudiation)
keyEncipherment	Wrap Key or Unwrap Key
dataEncipherment	Encrypt or Decrypt
keyAgreement	Key Agreement
keyCertSign	Certificate Sign

cRLSign	CRL Sign
encipherOnly	Encrypt
decipherOnly	Decrypt

Table 85: X.509 Key Usage to Cryptographic Usage Mask Mapping

Object	Encoding
Cryptographic Usage Mask	Integer

Table 86: 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, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects, Templates

Table 87: Cryptographic Usage Mask Attribute Rules

3.20 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.

Object	Encoding
Lease Time	Interval

Table 88: 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, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects

Table 89: Lease Time Attribute Rules

3.21 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 the three following fields:

- *Usage Limits Total* – the total number of Usage Limits Units 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 Count* – the currently remaining number of Usage Limits Units allowed to be protected by the object.
- *Usage Limits Unit* – The type of quantity for which this structure specifies a usage limit (e.g., byte, object).

When the attribute is initially set (usually during object creation or registration), the Usage Limits Count is set to the Usage Limits Total value allowed for the useful life of the object, and are decremented when the object is used. The server SHALL ignore the Usage Limits Count value if the attribute is specified in an operation that creates a new object. Changes made via the Modify Attribute operation reflect corrections to the Usage Limits Total value, but they SHALL NOT be changed once the Usage Limits Count value has changed by a Get Usage Allocation operation. The Usage Limits Count value 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	Long Integer	Yes
Usage Limits Count	Long Integer	Yes
Usage Limits Unit	Enumeration, see 9.1.3.2.31	Yes

Table 90: Usage Limits Attribute Structure

SHALL always have a value	No
Initially set by	Server (Total, Count, and Unit) or Client (Total and/or Unit only)
Modifiable by server	Yes
Modifiable by client	Yes (Total and/or Unit 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, Re-key Key Pair, Get Usage Allocation
Applies to Object Types	Keys, Templates

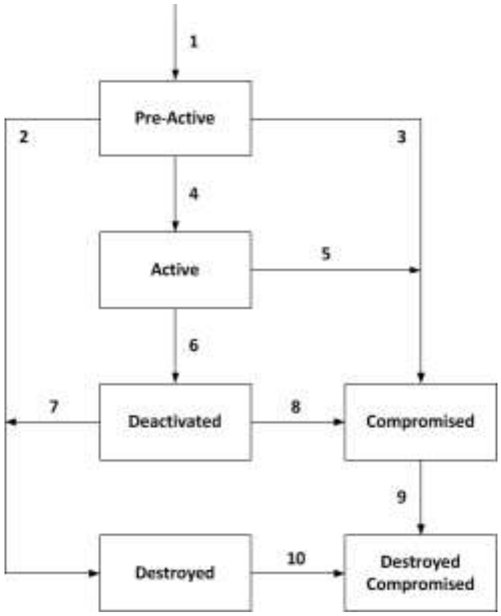
Table 91: Usage Limits Attribute Rules

3.22 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 **[SP800-57-1]**).

- *Pre-Active*: The object exists and SHALL NOT be used for any cryptographic purpose.
- *Active*: The object SHALL be transitioned to the *Active* state prior to being used for any cryptographic purpose. The object SHALL only be used for all cryptographic purposes that are allowed by its Cryptographic Usage Mask attribute. If a Process Start Date (see 3.25) attribute is set, then the object SHALL NOT be used for cryptographic purposes prior to the Process Start Date. If a Protect Stop Date (see 3.26) attribute is set, then the object SHALL NOT be used for cryptographic purposes after the Process Stop Date.
- *Deactivated*: The object SHALL NOT be used for applying cryptographic protection (e.g., encryption, signing, wrapping, MACing, deriving) . The object SHALL only be used for cryptographic purposes permitted by the Cryptographic Usage Mask attribute. The object SHOULD only be used to process cryptographically-protected information (e.g., decryption, signature verification, unwrapping, MAC verification under extraordinary circumstances and

Figure 1: Cryptographic Object States and Transitions



when special permission is granted.

- *Compromised*: The object SHALL NOT be used for applying cryptographic protection (e.g., encryption, signing, wrapping, MACing, deriving). The object SHOULD only be used to process cryptographically-protected information (e.g., decryption, signature verification, unwrapping, MAC verification in a client that is trusted to use managed objects that have been compromised. The object SHALL only be used for cryptographic purposes permitted by the Cryptographic Usage Mask attribute.
- *Destroyed*: The object SHALL NOT be used for any cryptographic purpose.
- *Destroyed Compromised*: The object SHALL NOT be used for any cryptographic purpose; however its compromised status SHOULD 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 value specified in the request, 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 successfully issues a Modify Attribute operation, modifying the Activation Date to a date in the past, or the current date, or
    - 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, or
    - The client successfully issues a Modify Attribute operation, modifying the Deactivation Date to a date in the past, or the current date.
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 9.1.3.2.18

Table 92: 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, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects

Table 93: State Attribute Rules

3.23 Initial Date

The *Initial Date* attribute contains 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 3.22). 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.

Object	Encoding
Initial Date	Date-Time

Table 94: 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, Re-key Key Pair
Applies to Object Types	All Objects

Table 95: Initial Date Attribute Rules

3.24 Activation Date

The *Activation Date* attribute contains the date and time when the Managed Cryptographic Object MAY begin to be used. This time corresponds to state transition 4 (see Section 3.22). 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.

Object	Encoding
Activation Date	Date-Time

Table 96: 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, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects, Templates

Table 97: Activation Date Attribute Rules

3.25 Process Start Date

The *Process Start Date* attribute 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.

Object	Encoding
Process Start Date	Date-Time

Table 98: 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 99: Process Start Date Attribute Rules

3.26 Protect Stop Date

The *Protect Stop Date* attribute is the date and time after which 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.

Object	Encoding
Protect Stop Date	Date-Time

Table 100: 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 101: Protect Stop Date Attribute Rules

3.27 Deactivation Date

The *Deactivation Date* attribute 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 3.22). This attribute SHALL NOT be changed or deleted before the object is destroyed, unless the object is in the Pre-Active or Active state.

Object	Encoding
Deactivation Date	Date-Time

Table 102: 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, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects, Templates

Table 103: Deactivation Date Attribute Rules

3.28 Destroy Date

The *Destroy Date* attribute is the date and time when the Managed Object was destroyed. This time corresponds to state transitions 2, 7, or 9 (see Section 3.22). 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.

Object	Encoding
Destroy Date	Date-Time

Table 104: 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 105: Destroy Date Attribute Rules

3.29 Compromise Occurrence Date

The *Compromise Occurrence Date* attribute 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 106: 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 107: Compromise Occurrence Date Attribute Rules

3.30 Compromise Date

The *Compromise Date* attribute contains 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 3.22). 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 code, or due to server policy or out-of-band administrative action.

Object	Encoding
Compromise Date	Date-Time

Table 108: 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 109: Compromise Date Attribute Rules

3.31 Revocation Reason

The *Revocation Reason* attribute is a structure (see Table 110) used to indicate why the Managed Cryptographic Object was revoked (e.g., “compromised”, “expired”, “no longer used”, etc.). This attribute is only set by the server as a part of the Revoke Operation.

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 9.1.3.2.19	Yes
Revocation Message	Text String	No

Table 110: 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 111: Revocation Reason Attribute Rules

3.32 Archive Date

The *Archive Date* attribute 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.

Object	Encoding
Archive Date	Date-Time

Table 112: 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

Table 113: Archive Date Attribute Rules

3.33 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. “default” is a reserved Text String for *Object Group*.

Object	Encoding
Object Group	Text String

Table 114: 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, Re-key Key Pair
Applies to Object Types	All Objects

Table 115: Object Group Attribute Rules

3.34 Fresh

The *Fresh* attribute is a Boolean attribute that indicates that the object has not yet been served to a client. The Fresh attribute SHALL be set to True when a new object is created on the server. The server SHALL change the attribute value to False as soon as the object has been served to a client.

Object	Encoding
Fresh	Boolean

Table 116: Fresh Attribute

SHALL always have a value	No
Initially set by	Client or 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, Re-key Key Pair, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects

Table 117: Fresh Attribute Rules

3.35 Link

The *Link* attribute is a structure (see Table 118) 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).

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 or Secret Data object: the object(s) from which the current symmetric key was derived.
- *Derived Key Link*: the symmetric key(s) or Secret Data object(s) that were derived from the current object.
- *Replacement Object Link:* For a Symmetric Key, an Asymmetric Private Key, or an Asymmetric Public Key object: the key that resulted from the re-key of the current key. For a Certificate object: the certificate that resulted from the re-certify. Note that there SHALL be only one such replacement object per Managed Object.
- *Replaced Object Link:* For a Symmetric Key, an Asymmetric Private Key, or an Asymmetric Public Key object: the key that was re-keyed to obtain the current key. For a Certificate object: the certificate that was re-certified to obtain the current certificate.
- *Parent Link:* For all object types: the owner, container or other parent object corresponding to the object.
- *Child Link:* For all object types: the subordinate, derived or other child object corresponding to the object.
- *Previous Link:* For all object types: the previous object to this object.
- *Next Link:* For all object types: the next object to this object.

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

Note that 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 9.1.3.2.20	Yes
Linked Object Identifier, see 3.1	Text String	Yes

Table 118: 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, Re-key Key Pair
Applies to Object Types	All Cryptographic Objects

Table 119: Link Attribute Structure Rules

3.36 Application Specific Information

The *Application Specific Information* attribute is a structure (see Table 120) 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 result in new Managed Object(s) on the server 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 4.25), then it SHALL return a suitable *Application Data* value. If the server does not support this namespace, then an error SHALL be returned.

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

Table 120: 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-key Key Pair, Re-certify
Applies to Object Types	All Objects

Table 121: Application Specific Information Attribute Rules

3.37 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 122: 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, Re-key Key Pair
Applies to Object Types	All Objects

Table 123: Contact Information Attribute Rules

3.38 Last Change Date

The *Last Change Date* attribute contains the date and time of the last change of the specified object.

Object	Encoding
Last Change Date	Date-Time

Table 124: 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, Re-key Key Pair, Add Attribute, Modify Attribute, Delete Attribute, Get Usage Allocation
Applies to Object Types	All Objects

Table 125: Last Change Date Attribute Rules

3.39 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.

Object	Encoding	
Custom Attribute	Any data type or structure. If a structure, then the structure SHALL NOT include sub structures	The name of the attribute SHALL start with 'x-' or 'y-'.

Table 126 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, Re-key Key Pair
Applies to Object Types	All Objects

Table 127: Custom Attribute Rules

3.40 Alternative Name

The *Alternative Name* attribute is used to identify and locate the object. This attribute is assigned by the client, and the *Alternative 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 alternative names. Clients are informed of such rules by a mechanism that is not specified by this standard. Alternative Names MAY NOT be unique within a given key management domain.

Object	Encoding	REQUIRED
Alternative Name	Structure	
Alternative Name Value	Text String	Yes
Alternative Name Type	Enumeration, see 9.1.3.2.34	Yes

Table 128: Alternative Name Attribute Structure

SHALL always have a value	No
Initially set by	Client
Modifiable by server	Yes (Only if no value present)
Modifiable by client	Yes
Deletable by client	Yes
Multiple instances permitted	Yes
Applies to Object Types	All Objects

Table 129: Alternative Name Attribute Rules

3.41 Key Value Present

Key Value Present is an OPTIONAL attribute of the managed object created by the server. It SHALL NOT be specified by the client in a Register request. *Key Value Present* SHALL be created by the server if the

Key Value is absent from the Key Block in a Register request. The value of Key Value Present SHALL NOT be modified by either the client or the server. *Key Value Present* attribute MAY be used as a part of the Locate operation. This attribute does not apply to Templates, Certificates, Public Keys or Opaque Objects.

Object	Encoding	REQUIRED
Key Value Present	Boolean	No

Table 130: Key Value Present 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	During Register operation
Applies to Object Types	Symmetric Key, Private Key, Split Key, Secret Data

Table 131: Key Value Present Attribute Rules

3.42 Key Value Location

Key Value Location is an OPTIONAL attribute of a managed object. It MAY be specified by the client when the Key Value is omitted from the Key Block in a Register request. *Key Value Location* is used to indicate the location of the Key Value absent from the object being registered. This attribute does not apply to Templates, Certificates, Public Keys or Opaque Objects.

Object	Encoding	REQUIRED
Key Value Location	Structure	
Key Value Location Value	Text String	Yes
Key Value Location Type	Enumeration, see 9.1.3.2.35	Yes

Table 132: Key Value Location Attribute

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	Never
Applies to Object Types	Symmetric Key, Private Key, Split Key, Secret Data

Table 133: Key Value Location Attribute Rules

3.43 Original Creation Date

The *Original Creation Date* attribute contains the date and time the object was originally created, which can be different from when the object is registered with a key management server.

It is OPTIONAL for an object being registered by a client. The *Original Creation Date* MAY be set by the client during a Register operation. If no *Original Creation Date* attribute was set by the client during a Register operation, it MAY do so at a later time through an Add Attribute operation for that object.

It is mandatory for an object created on the server as a result of a Create, Create Key Pair, Derive Key, Re-key, or Re-key Key Pair operation. In such cases the *Original Creation Date* SHALL be set by the server and SHALL be the same as the *Initial Date* attribute.

In all cases, once the *Original Creation Date* is set, it SHALL NOT be deleted or updated.

Object	Encoding
Original Creation Date	Date-Time

Table 134: Original Creation Date Attribute

SHALL always have a value	No
Initially set by	Client or Server (when object is generated 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, Derive Key, Re-key, Re-key Key Pair
Applies to Object Types	All Objects

Table 135: Original Creation Date 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.

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.

The ID Placeholder is obtained from the Unique Identifier returned in response to the Create, Create Pair, Register, Derive Key, Re-key, Re-key Key Pair, 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-instance attributes, the union of attribute values is used when the attributes are specified more than once.

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 4.23) before they MAY be specified (i.e., as on-line objects).

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).

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 attributes using to be associated with the new object templates and/or individual attributes.

Table 136: Create Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Type of object created.
Unique Identifier, see 3.1	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.

Table 137: Create Response Payload

Table 138 indicates which attributes SHALL be included in the Create request using the Template-Attribute object.

Attribute	REQUIRED
Cryptographic Algorithm, see 3.4	Yes
Cryptographic Usage Mask, see 3.19	Yes

Table 138: 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.

For the Private Key, the server SHALL create a Link attribute of Link Type Public Key pointing to the Public Key. For the Public Key, the server SHALL create a Link attribute of Link Type Private Key pointing to the Private Key. 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	No	Specifies desired attributes in templates and/or as individual attributes to be associated with the new object that apply to both the Private and Public Key Objects.
Private Key Template-Attribute, see 2.1.8	No	Specifies templates and/or attributes to be associated with the new object that apply to the Private Key Object. Order of precedence applies.
Public Key Template-Attribute, see 2.1.8	No	Specifies templates and/or attributes to be associated with the new object that apply to the Public Key Object. Order of precedence applies.

Table 139: 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 SHALL be 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 last value of the single-instance attribute that conflicts takes precedence.

Response Payload		
Object	**REQUIRED**	**Description**
Private Key Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly created Private Key object.
Public Key Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly created Public Key object.
Private Key Template-Attribute, see 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	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 140: Create Key Pair Response Payload

Table 141 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	No	Yes
Cryptographic Usage Mask, see 3.19	Yes	No
Cryptographic Domain Parameters, see 3.7	No	Yes
Cryptographic Parameters, see 3.6	No	Yes

Table 141: Create Key Pair Attribute Requirements

Setting the same Cryptographic Length value for both private and public key does not imply that both keys are of equal length. For RSA, Cryptographic Length corresponds to the bit length of the Modulus. For DSA and DH algorithms, Cryptographic Length corresponds to the bit length of parameter P, and the bit length of Q is set separately in the Cryptographic Domain Parameters attribute. For ECDSA, ECDH, and ECMQV algorithms, Cryptographic Length corresponds to the bit length of parameter Q.

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 contain the object itself for storage by the server.

The request contains information about the type of object being registered and attributes to be assigned to the object (e.g., Cryptographic Algorithm, Cryptographic Length, etc.). This information SHALL 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 to be associated with the new object using templates and/or individual attributes.
Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template Secret Data or Opaque Object, see 2.2	Yes	The object being registered. The object and attributes MAY be wrapped.

Table 142: Register Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 143: 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.
Certificate Length, see 3.9	Yes. Only applies to Certificates.
Cryptographic Usage Mask, see 3.19	Yes.
Digital Signature Algorithm, see 3.16	Yes, MAY be omitted only if this information is encapsulated in the Certificate object. Only applies to Certificates.

Table 144: 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 146.

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

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

For the existing key, the server SHALL create a Link attribute of Link Type Replacement Object pointing to the replacement key. For the replacement key, the server SHALL create a Link attribute of Link Type Replaced Key pointing to the existing key.

An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date of the replacement key. If no Offset is specified, the Activation Date, Process Start Date, Protect Stop Date and Deactivation Date values are copied from the existing 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 145: Computing New Dates from Offset during Re-key

Attributes requiring special handling when creating the replacement key are:

Attribute	Action
Initial Date, see 3.23	Set to the current time
Destroy Date, see 3.28	Not set
Compromise Occurrence Date, see 3.29	Not set
Compromise Date, see 3.30	Not set
Revocation Reason, see 3.31	Not set
Unique Identifier, see 3.1	New value generated
Usage Limits, see 3.21	The Total value is copied from the existing key, and the Count value in the existing key is set to the Total value.
Name, see 3.2	Set to the name(s) of the existing key; all name attributes are removed from the existing key.
State, see 3.22	Set based on attributes values, such as dates, as shown in Table 145
Digest, see 3.16	Recomputed from the replacement key value
Link, see 3.35	Set to point to the existing key as the replaced key
Last Change Date, see 3.38	Set to current time

Table 146: Re-key Attribute Requirements

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 147: Re-key Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 148: Re-key Response Payload

4.5 Re-key Key Pair

This request is used to generate a replacement key pair for an existing public/private key pair. It is analogous to the Create Key Pair operation, except that attributes of the replacement key pair are copied from the existing key pair, with the exception of the attributes listed in Table 150.

As the replacement of the key pair takes over the name attribute for the existing public/private key pair, Re-key Key Pair SHOULD only be performed once on a given key pair.

For both the existing public key and private key, the server SHALL create a Link attribute of Link Type Replacement Key pointing to the replacement public and private key, respectively. For both the replacement public and private key, the server SHALL create a Link attribute of Link Type Replaced Key pointing to the existing public and private key, respectively.

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

An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date of the replacement key pair. If no Offset is specified, the Activation Date and Deactivation Date values are copied from the existing key pair. If Offset is set and dates exist for the existing key pair, then the dates of the replacement key pair SHALL be set based on the dates of the existing key pair as follows

Attribute in Existing Key Pair	Attribute in Replacement Key Pair
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 149: Computing New Dates from Offset during Re-key Key Pair

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

Attribute	Action
Private Key Unique Identifier, see 3.1	New value generated
Public Key Unique Identifier, see 3.1	New value generated
Name, see 3.2	Set to the name(s) of the existing public/private keys; all name attributes of the existing public/private keys are removed.
Digest, see 3.17	Recomputed for both replacement public and private keys from the new public and private key values
Usage Limits, see 3.21	The Total Bytes/Total Objects value is copied from the existing key pair, while the Byte Count/Object Count values are set to the Total Bytes/Total Objects.
State, see 3.22	Set based on attributes values, such as dates, as shown in Table 149.
Initial Date, see 3.23	Set to the current time
Destroy Date, see 3.28	Not set
Compromise Occurrence Date, see 3.29	Not set
Compromise Date, see 3.30	Not set
Revocation Reason, see 3.31	Not set
Link, see 3.35	Set to point to the existing public/private keys as the replaced public/private keys
Last Change Date, see 3.38	Set to current time

Table 150: Re-key Key Pair Attribute Requirements

Request Payload		
Object	**REQUIRED**	**Description**
Private Key Unique Identifier, see 3.1	No	Determines the existing Asymmetric key pair to be re-keyed. If omitted, then the ID Placeholder is substituted by the server.
Offset	No	An Interval object indicating the difference between the Initialization date and the Activation Date of the replacement key pair to be created.
Common Template-Attribute, see 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	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	No	Specifies templates and/or attributes that apply to the Public Key Object. Order of precedence applies.

Table 151: Re-key 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 value of the single-instance attribute takes precedence.

Response Payload		
Object	**REQUIRED**	**Description**
Private Key Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly created replacement Private Key object.
Public Key Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly created replacement Public Key object.
Private Key Template-Attribute, see 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	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 152: Re-key Key Pair Response Payload

4.6 Derive Key

This request is used to derive a Symmetric Key or Secret Data object from keys or Secret Data objects that are 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. If the specified length exceeds the output of the derivation method, then the server SHALL return an error.

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 use 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:

- *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.

For the keys or Secret Data objects from which the key or Secret Data object is derived, the server SHALL create a Link attribute of Link Type Derived Key pointing to the Symmetric Key or Secret Data object derived as a result of this operation. For the Symmetric Key or Secret Data object derived as a result of this operation, the server SHALL create a Link attribute of Link Type Derivation Base Object pointing to the keys or Secret Data objects from which the key or Secret Data object is derived.

Request Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Determines the type of object to be created.
Unique Identifier, see 3.1	Yes. MAY be repeated	Determines the object or objects to be used to derive a new key. 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 9.1.3.2.21	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 attributes to be associated with the new object using templates and/or individual attributes; the length and algorithm SHALL always be specified for the creation of a symmetric key.

Table 153: Derive Key Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 154: Derive Key Response Payload

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

Table 155: 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.

The PBKDF2 derivation method takes 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 (if different than those defined in **[PKCS#5]**) 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.

Table 156: PBKDF2 Derivation Parameters Structure

4.7 Certify

This request is used to generate a Certificate object for a public key. This request supports the certification of a new public key, as well as the 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. If the server does not support this operation, an error SHALL be returned.

The Certificate Request object MAY be omitted, in which case the public key for which a Certificate object is generated SHALL be specified by its Unique Identifier only. If the Certificate Request Type and the Certificate Request objects are omitted from the request, then the Certificate Type SHALL be specified using the Template-Attribute object.

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.) to be submitted to the server.

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.

For the public key, the server SHALL create a Link attribute of Link Type Certificate pointing to the generated certificate. For the generated certificate, the server SHALL create a Link attribute of Link Type Public Key pointing to the Public Key.

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 3.1	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 9.1.3.2.22	No	An Enumeration object specifying the type of certificate request. It is REQUIRED if the Certificate Request is present.
Certificate Request	No	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.

Table 157: Certify Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 158: Certify Response Payload

4.8 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. If the server does not support this operation, an error SHALL be returned.

The Certificate Request object MAY be omitted, in which case the public key for which a Certificate object is generated SHALL be specified by its Unique Identifier only. If the Certificate Request Type and the Certificate Request objects are omitted and the Certificate Type is not specified using the Template-Attribute object in the request, then the Certificate Type of the new certificate SHALL be the same as that of the existing certificate.

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.) to be submitted to the server.

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.

For the existing certificate, the server SHALL create a Link attribute of Link Type Replacement pointing to the new certificate. For the new certificate, the server SHALL create a Link attribute of Link Type Replaced pointing to the existing certificate. For the public key, the server SHALL change the Link attribute of Link Type Certificate 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 no Offset is specified, the Activation Date and Deactivation Date values are copied from the existing certificate. If Offset is set and dates exist for the existing certificate, then the dates of the new certificate SHALL be set based on the dates of the existing certificate 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 159: 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 3.23	Set to current time.
Destroy Date, see 3.28	Not set.
Revocation Reason, see 3.31	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 3.22	Set based on attributes values, such as dates, as shown in Table 159.
Digest, see 3.16	Recomputed from the new certificate value.
Link, see 3.35	Set to point to the existing certificate as the replaced certificate.
Last Change Date, see 3.38	Set to current time.

Table 160: Re-certify Attribute Requirements

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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 9.1.3.2.22	No	An Enumeration object specifying the type of certificate request. It is REQUIRED if the Certificate Request is present.
Certificate Request	No	A Byte String object with the certificate request.
Offset	No	An Interval object indicating the difference between the Initial Date 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.

Table 161: Re-certify Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 162: Re-certify Response Payload

4.9 Locate

This operation requests that the server search for one or more Managed Objects, depending on the attributes specified in the request. 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 server. The request MAY 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.

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. If no attribute is specified in the request, any object SHALL be deemed to match the Locate request.

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.

Wild-cards or regular expressions (defined, e.g., in **[ISO/IEC 9945-2]**) MAY be supported by specific key management system implementations for matching attribute fields when the field type is a Text String or a Byte String.

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.

When the Usage Limits attribute is specified in the request, matching candidate objects SHALL have a Usage Limits Count and Usage Limits Total 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.

When the Object Group attribute and the Object Group Member flag are specified in the request, and the value specified for Object Group Member is ‘Group Member Fresh’, matching candidate objects SHALL be fresh objects (see 3.34) from the object group. If there are no more fresh objects in the group, the server MAY choose to generate a new object on-the-fly, based on server policy. If the value specified for Object Group Member is ‘Group Member Default’, the server locates the default object as defined by server policy.

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.

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.
Object Group Member, see 9.1.3.2.33	No	An Enumeration object that indicates the object group member type.
Attribute, see 3	No, MAY be	Specifies an attribute and its value(s)

	repeated	that are REQUIRED to match those in a candidate object (according to the matching rules defined above).

Table 163: Locate Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No, MAY be repeated	The Unique Identifier of the located objects.

Table 164: Locate Response Payload

4.10 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, Re-key or Re-key Key Pair operation, and followed by a Get operation.

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.

In a batch containing a Check operation the Batch Order Option SHOULD be set to true. Only STOP or UNDO Batch Error Continuation Option values SHOULD be used by the client in such a batch. Additional attributes that MAY be specified in the request are limited to:

- Usage Limits Count (see Section 3.21) – The request MAY contain the usage amount that the client deems necessary to complete its needed function. This does not require that any subsequent Get Usage Allocation operations request this amount. It only means that the client is ensuring that the amount specified is available.
- Cryptographic Usage Mask – This is used to specify the cryptographic operations for which the client intends to use the object (see Section 3.19). 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.
- Lease Time – This specifies a desired lease time (see Section 3.20). 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.

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

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	Determines the object being checked. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Usage Limits Count, see 3.21	No	Specifies the number of Usage Limits Units to be protected to be checked against server policy.
Cryptographic Usage Mask, see 3.19	No	Specifies the Cryptographic Usage for which the client intends to use the object.
Lease Time, see 3.20	No	Specifies a Lease Time value that the Client is asking the server to validate against server policy.

Table 165: Check Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes, unless a failure,	The Unique Identifier of the object.
Usage Limits Count, see 3.21	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.
Cryptographic Usage Mask, see 3.19	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 3.20	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.

Table 166: Check Response Payload

4.11 Get

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

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:

- 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 be supported by the server.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 167: Get Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Type of object.
Unique Identifier, see 3.1	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.

Table 168: Get Response Payload

4.12 Get Attributes

This operation requests one or more attributes associated with 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. If no attribute name is specified in the request, all attributes SHALL be deemed to match the Get Attributes request. The same attribute name SHALL NOT be present more than once in a request.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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	No, MAY be repeated	Specifies the name of an attribute associated with the object.

Table 169: Get Attributes Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Attribute, see 2.1.1	No, MAY be repeated	The requested attribute associated with the object.

Table 170: Get Attributes Response Payload

4.13 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.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 171: Get Attribute List Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Attribute Name, see 2.1.1	Yes, MAY be repeated	The names of the available attributes associated with the object.

Table 172: Get Attribute List Response Payload

4.14 Add Attribute

This operation requests the server to add a new attribute instance to be associated with a Managed Object and set 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 single-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. The Attribute Index SHALL NOT be specified in the request. The response returns a new Attribute Index and 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.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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 as an attribute for the object.

Table 173: Add Attribute Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Attribute, see 2.1.1	Yes	The added attribute associated with the object.

Table 174: Add Attribute Response Payload

4.15 Modify Attribute

This operation requests the server to modify 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, the attribute name, the OPTIONAL Attribute Index, and the new value. If no Attribute Index is specified in the request, then the Attribute Index SHALL be assumed to be 0. Only existing attributes MAY be changed via this operation. New attributes SHALL only be added by the Add Attribute operation. Only the specified instance of the attribute SHALL be modified. Specifying an Attribute Index for which there exists no Attribute object SHALL result in an error. The response returns the modified Attribute (new value) and the Attribute Index MAY be omitted if the index of the modified attribute instance is 0. Multiple Modify Attribute requests MAY be included in a single batched request to modify multiple attributes.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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 associated with the object to be modified.

Table 175: Modify Attribute Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Attribute, see 2.1.1	Yes	The modified attribute associated with the object with the new value.

Table 176: Modify Attribute Response Payload

4.16 Delete Attribute

This operation requests the server to delete 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 the OPTIONAL Attribute Index of the attribute. If no Attribute Index is specified in the request, then the Attribute Index SHALL be assumed to be 0. Attributes that are always REQUIRED to have a value SHALL never be deleted by this operation. Attempting to delete a non-existent attribute or specifying an Attribute Index for which there exists no Attribute Value SHALL result in an error. The response returns the deleted Attribute and the Attribute Index MAY be omitted if the index of the deleted attribute instance is 0. Multiple Delete Attribute requests MAY be included in a single batched request to delete multiple attributes.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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 associated with the object to be deleted.
Attribute Index, see 2.1.1	No	Specifies the Index of the Attribute.

Table 177: Delete Attribute Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Attribute, see 2.1.1	Yes	The deleted attribute associated with the object.

Table 178: Delete Attribute Response Payload

4.17 Obtain Lease

This operation requests the server 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.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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.

Table 179: Obtain Lease Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Lease Time, see 3.20	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 3.38	Yes	The date and time indicating when the

		latest change was made to the contents or any attribute of the specified object.

Table 180: Obtain Lease Response Payload

4.18 Get Usage Allocation

This operation requests the server to obtain an allocation from the current Usage Limits value 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 field in the request specifies the number of units that the client needs to protect. 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 is going to be 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 3.1	No	Determines the object whose usage allocation is being requested. If omitted, then the ID Placeholder is substituted by the server.
Usage Limits Count, see Usage Limits Count field in 3.21	Yes	The number of Usage Limits Units to be protected.

Table 181: Get Usage Allocation Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.

Table 182: Get Usage Allocation Response Payload

4.19 Activate

This operation requests the server 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.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	Determines the object being activated. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Table 183: Activate Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.

Table 184: Activate Response Payload

4.20 Revoke

This operation requests the server 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., “key compromise”, “cessation of operation”, etc.). Special authentication and authorization SHOULD be enforced to perform this request (see **[KMIP-UG]**). Only the object owner or an authorized security officer SHOULD be allowed to issue this request. The operation has one of two effects. If the revocation reason is “key compromise”, 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.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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 3.31	Yes	Specifies the reason for revocation.
Compromise Occurrence Date, see 3.29	No	SHALL be specified if the Revocation Reason is 'key compromise'.

Table 185: Revoke Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.

Table 186: Revoke Response Payload

4.21 Destroy

This operation 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 owner 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. Cryptographic Objects MAY only be destroyed if they are in either Pre-Active or Deactivated state. A Cryptographic Object in the Active state MAY be destroyed if the server sets the Deactivation date (the state of the object transitions to Deactivated) to a date that is prior to or equal to the current date before destroying the object.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	Determines the object being destroyed. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Table 187: Destroy Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.

Table 188: Destroy Response Payload

4.22 Archive

This operation 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 owner 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, the key management system MAY archive the object.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	Determines the object being archived. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Table 189: Archive Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.

Table 190: Archive Response Payload

4.23 Recover

This operation is used to obtain access to a Managed Object that has been archived. This request MAY need 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]**).

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	Determines the object being

		recovered. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

Table 191: Recover Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.

Table 192: Recover Response Payload

4.24 Validate

This operation requests the server to 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.

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 3.1	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. If omitted, the current date and time SHALL be assumed.

Table 193: Validate Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Validity Indicator, see 9.1.3.2.23	Yes	An Enumeration object indicating whether the certificate chain is valid, invalid, or unknown.

Table 194: Validate Response Payload

4.25 Query

This operation 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
- Query Extension List
- Query Extension Map
- Query Attestation Types

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 3.36). These fields SHALL only be returned in the response if the request contains a Query Application Namespaces value in the Query Function field.

The *Extension Information* fields in the response contain the descriptions of Objects with Item Tag values in the Extensions range that are supported by the server (see Section 2.1.9). If the request contains a *Query Extension List* and/or *Query Extension Map* value in the Query Function field, then the Extensions Information fields SHALL be returned in the response. If the Query Function field contains the Query Extension Map value, then the Extension Tag and Extension Type fields SHALL be specified in the Extension Information values.

The *Attestation Type* fields in the response contain Attestation Type enumerated values, which SHALL list all the attestation types that the server supports. If the request contains a *Query Attestation Types* value in the Query Function field, then this field SHALL be returned in the response if the server supports any Attestation Types.

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

Request Payload		
Object	**REQUIRED**	**Description**
Query Function, see 9.1.3.2.24	Yes, MAY be Repeated	Determines the information being queried.

Table 195: Query Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Operation, see 9.1.3.2.27	No, MAY be repeated	Specifies an Operation that is supported by the server.
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 3.36	No, MAY be repeated	Specifies an Application Namespace supported by the server.
Extension Information, see 2.1.9	No, MAY be repeated	SHALL be returned if Query Extension List or Query Extension Map is requested and supported by the server.
Attestation Type, see 9.1.3.2.36	No, MAY be repeated	Specifies an Attestation Type that is supported by the server.

Table 196: Query Response Payload

4.26 Discover Versions

This operation is used by the client to determine a list of protocol versions that is supported by the server. The request payload contains an OPTIONAL list of protocol versions that is supported by the client. The protocol versions SHALL be ranked in order of preference (highest preference first).

The response payload contains a list of protocol versions that are supported by the server. The protocol versions are ranked in order of preference (highest preference first). If the client provides the server with a list of supported protocol versions in the request payload, the server SHALL return only the protocol versions that are supported by both the client and server. The server SHOULD list all the protocol versions supported by both client and server. If the protocol version specified in the request header is not specified in the request payload and the server does not support any protocol version specified in the request payload, the server SHALL return an empty list in the response payload. If no protocol versions are specified in the request payload, the server SHOULD return all the protocol versions that are supported by the server.

Request Payload		
Object	**REQUIRED**	**Description**
Protocol Version, see 6.1	No, MAY be Repeated	The list of protocol versions supported by the client ordered in highest preference first.

Table 197: Discover Versions Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Protocol Version, see 6.1	No, MAY be repeated	The list of protocol versions supported by the server ordered in highest preference first.

Table 198: Discover Versions Response Payload

4.27 Cancel

This operation requests the server 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:

- *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.

Table 199: Cancel Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Asynchronous Correlation Value, see 6.8	Yes	Specified in the request.
Cancellation Result, see 9.1.3.2.25	Yes	Enumeration indicating the result of the cancellation.

Table 200: Cancel Response Payload

4.28 Poll

This operation 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.

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

Table 201: 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.

4.29 Encrypt

This operation requests the server to perform an encryption operation on the provided data using a Managed Cryptographic Object as the key for the encryption operation.

The request contains information about the cryptographic parameters (mode and padding method), the data to be encrypted, and the IV/Counter/Nonce to use. The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object. The IV/Counter/Nonce MAY also be omitted from the request if the cryptographic parameters indicate that the server shall generate a Random IV on behalf of the client or the encryption algorithm does not need an IV/Counter/Nonce. The server does not store or otherwise manage the IV/Counter/Nonce.

If the Managed Cryptographic Object referenced has a Usage Limits attribute then the server SHALL obtain an allocation from the current Usage Limits value prior to performing the encryption operation. If the allocation is unable to be obtained the operation SHALL return with a result status of Operation Failed and result reason of Permission Denied.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the encryption operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	The Unique Identifier of the Managed Cryptographic Object that is the key to use for the encryption operation. If omitted, then the ID Placeholder value SHALL be used by the server as the Unique Identifier.
Cryptographic Parameters, see 3.6	No	The Cryptographic Parameters (Block Cipher Mode, Padding Method, RandomIV) corresponding to the particular encryption method requested. If omitted then the Cryptographic Parameters associated with the Managed Cryptographic Object with the lowest Attribute Index SHALL be used. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed.
Data	Yes	The data to be encrypted (as a Byte String).
IV/Counter/Nonce	No	The initialization vector, counter or nonce to be used (where appropriate).

Table 202: Encrypt Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the Managed Cryptographic Object that was the key used for the encryption operation.
Data	Yes	The encrypted data (as a Byte String).
IV/Counter/Nonce	No	The value used if the Cryptographic Parameters specified Random IV and

		the IV/Counter/Nonce value was not provided in the request and the algorithm requires the provision of an IV/Counter/Nonce.

Table 203: Encrypt Response Payload

4.30 Decrypt

This operation requests the server to perform a decryption operation on the provided data using a Managed Cryptographic Object as the key for the decryption operation.

The request contains information about the cryptographic parameters (mode and padding method), the data to be decrypted, and the IV/Counter/Nonce to use. The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object. The initialization vector/counter/nonce MAY also be omitted from the request if the algorithm does not use an IV/Counter/Nonce.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the decryption operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	The Unique Identifier of the Managed Cryptographic Object that is the key to use for the decryption operation. If omitted, then the ID Placeholder value SHALL be used by the server as the Unique Identifier.
Cryptographic Parameters, see 3.6	No	The Cryptographic Parameters (Block Cipher Mode, Padding Method) corresponding to the particular decryption method requested. If omitted then the Cryptographic Parameters associated with the Managed Cryptographic Object with the lowest Attribute Index SHALL be used. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed.
Data	Yes	The data to be decrypted (as a Byte String).
IV/Counter/Nonce	No	The initialization vector, counter or nonce to be used (where appropriate).

Table 204: Decrypt Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the Managed Cryptographic Object that is the key used for the decryption operation.
Data	Yes	The decrypted data (as a Byte String).

Table 205: Decrypt Response Payload

4.31 Sign

This operation requests the server to perform a signature operation on the provided data using a Managed Cryptographic Object as the key for the signature operation.

The request contains information about the cryptographic parameters (digital signature algorithm or cryptographic algorithm and hash algorithm) and the data to be signed. The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object.

If the Managed Cryptographic Object referenced has a Usage Limits attribute then the server SHALL obtain an allocation from the current Usage Limits value prior to performing the signing operation. If the allocation is unable to be obtained the operation SHALL return with a result status of Operation Failed and result reason of Permission Denied.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the signature operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	The Unique Identifier of the Managed Cryptographic Object that is the key to use for the signature operation. If omitted, then the ID Placeholder value SHALL be used by the server as the Unique Identifier.
Cryptographic Parameters, see 3.6	No	The Cryptographic Parameters (Digital Signature Algorithm or Cryptographic Algorithm and Hashing Algorithm) corresponding to the particular signature generation method requested. If omitted then the Cryptographic Parameters associated with the Managed Cryptographic Object with the lowest Attribute Index SHALL be used. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a

		Result Status of Operation Failed.
Data	Yes	The data to be signed (as a Byte String).

Table 206: Sign Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the Managed Cryptographic Object that is the key used for the signature operation.
Signature Data	Yes	The signed data (as a Byte String).

Table 207: Sign Response Payload

4.32 Signature Verify

This operation requests the server to perform a signature verify operation on the provided data using a Managed Cryptographic Object as the key for the signature verification operation.

The request contains information about the cryptographic parameters (digital signature algorithm or cryptographic algorithm and hash algorithm) and the signature to be verified and MAY contain the data that was passed to the signing operation (for those algorithms which need the original data to verify a signature).

The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the OPTIONAL data recovered from the signature (for those signature algorithms where data recovery from the signature is supported). The validity of the signature is indicated by the Validity Indicator field.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	The Unique Identifier of the Managed Cryptographic Object that is the key to use for the signature verify operation. If omitted, then the ID Placeholder value SHALL be used by the server as the Unique Identifier.
Cryptographic Parameters, see 3.6	No	The Cryptographic Parameters (Digital Signature Algorithm or Cryptographic Algorithm and Hashing Algorithm) corresponding to the particular signature verification method requested. If omitted then the Cryptographic Parameters associated with the Managed Cryptographic Object with the lowest Attribute Index

		SHALL be used. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed.
Data	No	The data that was signed (as a Byte String).
Signature Data	Yes	The signature to be verified (as a Byte String).

1794 *Table 208: Signature Verify Request Payload*

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the Managed Cryptographic Object that is the key used for the verification operation.
Validity Indicator, see 9.1.3.2.23	Yes	An Enumeration object indicating whether the signature is valid, invalid, or unknown.
Data	No	The OPTIONAL recovered data (as a Byte String) for those signature algorithms where data recovery from the signature is supported.

1795 *Table 209: Signature Verify Response Payload*

1796 4.33 MAC

1797 This operation requests the server to perform message authentication code (MAC) operation on the
1798 provided data using a Managed Cryptographic Object as the key for the MAC operation.

1799 The request contains information about the cryptographic parameters (cryptographic algorithm) and the
1800 data to be MACed. The cryptographic parameters MAY be omitted from the request as they can be
1801 specified as associated attributes of the Managed Cryptographic Object.

1802 The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and
1803 the result of the MAC operation.

1804 The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason)
1805 in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	The Unique Identifier of the Managed Cryptographic Object that is the key to use for the MAC operation. If omitted, then the ID Placeholder value SHALL be used by the server as the Unique

		Identifier.
Cryptographic Parameters, see 3.6	No	The Cryptographic Parameters (Cryptographic Algorithm) corresponding to the particular MAC method requested. If omitted then the Cryptographic Parameters associated with the Managed Cryptographic Object with the lowest Attribute Index SHALL be used. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed.
Data	Yes	The data to be MACed (as a Byte String).

Table 210: MAC Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the Managed Cryptographic Object that is the key used for the MAC operation.
MAC Data	Yes	The data MACed (as a Byte String).

Table 211: MAC Response Payload

4.34 MAC Verify

This operation requests the server to perform message authentication code (MAC) verify operation on the provided data using a Managed Cryptographic Object as the key for the MAC verify operation.

The request contains information about the cryptographic parameters (cryptographic algorithm) and the data to be MAC verified and MAY contain the data that was passed to the MAC operation (for those algorithms which need the original data to verify a MAC). The cryptographic parameters MAY be omitted from the request as they can be specified as associated attributes of the Managed Cryptographic Object.

The response contains the Unique Identifier of the Managed Cryptographic Object used as the key and the result of the MAC verify operation. The validity of the MAC is indicated by the Validity Indicator field.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	No	The Unique Identifier of the Managed Cryptographic Object that is the key to use for the MAC verify operation. If omitted, then the ID Placeholder value SHALL be used by the server as the

		Unique Identifier.
Cryptographic Parameters, see 3.6	No	The Cryptographic Parameters (Cryptographic Algorithm) corresponding to the particular MAC method requested. If omitted then the Cryptographic Parameters associated with the Managed Cryptographic Object with the lowest Attribute Index SHALL be used. If there are no Cryptographic Parameters associated with the Managed Cryptographic Object and the algorithm requires parameters then the operation SHALL return with a Result Status of Operation Failed.
Data	No	The data that was MACed (as a Byte String).
MAC Data	Yes	The data to be MAC verified (as a Byte String).

Table 212: MAC Verify Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	Yes	The Unique Identifier of the Managed Cryptographic Object that is the key used for the verification operation.
Validity Indicator, see 9.1.3.2.23	Yes	An Enumeration object indicating whether the MAC is valid, invalid, or unknown.

Table 213: MAC Verify Response Payload

4.35 RNG Retrieve

This operation requests the server to return output from a Random Number Generator (RNG).

The request contains the quantity of output requested.

The response contains the RNG output.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Data Length	Yes	The amount of random number generator output to be returned (in bytes).

Table 214: RNG Retrieve Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Data	Yes	The random number generator output.

Table 215: RNG Retrieve Response Payload

4.36 RNG Seed

This operation requests the server to seed a Random Number Generator.

The request contains the seeding material.

The response contains the amount of seed data used.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

The server MAY elect to ignore the information provided by the client (i.e. not accept the seeding material) and MAY indicate this to the client by returning zero as the value in the Data Length response. A client SHALL NOT consider a response from a server which does not use the provided data as an error.

Request Payload		
Object	**REQUIRED**	**Description**
Data	Yes	The data to be provided as a seed to the random number generator.

Table 216: RNG Seed Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Data Length	Yes	The amount of seed data used (in bytes).

Table 217: RNG Seed Response Payload

4.37 Hash

This operation requests the server to perform a hash operation on the data provided.

The request contains information about the cryptographic parameters (hash algorithm) and the data to be hashed.

The response contains the result of the hash operation.

The success or failure of the operation is indicated by the Result Status (and if failure the Result Reason) in the response header.

Request Payload		
Object	**REQUIRED**	**Description**
Cryptographic Parameters, see 3.6	Yes	The Cryptographic Parameters (Hashing Algorithm) corresponding to the particular hash method requested.
Data	Yes	The data to be hashed (as a Byte String).

Table 218: MAC Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Data	Yes	The hashed data (as a Byte String).

Table 219: HASH Response Payload

4.38 Create Split Key

This operation requests the server to generate a new split key and register all the splits as individual new Managed Cryptographic Objects.

The request contains attributes to be assigned to the objects (e.g., Split Key Parts, Split Key Threshold, Split Key Method, Cryptographic Algorithm, Cryptographic Length, etc.). The request MAY contain the Unique Identifier of an existing cryptographic object that the client requests be split by the server. If the attributes supplied in the request do not match those of the key supplied, the attributes of the key take precedence.

The response contains the Unique Identifiers of all created objects. The ID Placeholder value SHALL be set to the Unique Identifier of the split whose Key Part Identifier is 1.

Request Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Determines the type of object to be created.
Unique Identifier, see 3.1	No	The Unique Identifier of the key to be split (if applicable).
Split Key Parts	Yes	The total number of parts.
Split Key Threshold	Yes	The minimum number of parts needed to reconstruct the entire key.
Split Key Method	Yes	
Prime Field Size	No	
Template-Attribute, see 2.1.8	Yes	Specifies desired object attributes using templates and/or individual attributes.

Table 220: Create Split Key Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Type of object created.
Unique Identifier, see 3.1	Yes, MAY be repeated	The list of Unique Identifiers of the newly created objects.
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 system.

Table 221: Create Split Key Response Payload

4.39 Join Split Key

This operation requests the server to combine a list of Split Keys into a single Managed Cryptographic Object. The number of Unique Identifiers in the request SHALL be at least the value of the Split Key Threshold defined in the Split Keys.

The request contains the Object Type of the Managed Cryptographic Object that the client requests the Split Key Objects be combined to form. If the Object Type formed is Secret Data, the client MAY include the Secret Data Type in the request.

The response contains the Unique Identifier of the object obtained by combining the Split Keys. 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.
Unique Identifier, see 3.1	Yes, MAY be repeated	Determines the Split Keys to be combined to form the object returned by the server. The minimum number of identifiers is specified by the Split Key Threshold field in each of the Split Keys.
Secret Data Type	No	Determines which Secret Data type the Split Keys form.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

Table 222: Join Split Key Request Payload

Response Payload		
Object	**REQUIRED**	**Description**
Object Type, see 3.3	Yes	Type of object created.
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object obtained by combining the Split Keys.
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 system.

Table 223: Join Split Key Response Payload

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 243), 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 244) 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.

Message Payload		
Object	**REQUIRED**	**Description**
Unique Identifier, see 3.1	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. In case an attribute was deleted, the Attribute structure (see 2.1.1) in question SHALL NOT contain the Attribute Value field.

Table 224: 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 243), 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 244) 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.

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. In case the object with the Replaced Unique Identifier does not exist at the client, the client SHALL interpret this as if the Put Function contained the value New.

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 3.1	Yes	The Unique Identifier of the object.
Put Function, see 9.1.3.2.26	Yes	Indicates function for Put message.
Replaced Unique Identifier, see 3.1	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.

Table 225: 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 SHALL be 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.

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

Table 226: 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.

Object	Encoding
Operation	Enumeration, see 9.1.3.2.27

Table 227: Operation in Batch Item

6.3 Maximum Response Size

This is an OPTIONAL field contained in a request message, and is used to indicate the maximum size of a response, in bytes, that the requester SHALL be able to handle. It SHOULD only be sent in requests that possibly return large replies.

Object	Encoding
Maximum Response Size	Integer

Table 228: Maximum Response Size in Message Request Header

6.4 Unique Batch Item ID

This is an OPTIONAL field 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 229: Unique Batch Item ID in Batch Item

6.5 Time Stamp

This is an OPTIONAL field 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.

Object	Encoding
Time Stamp	Date-Time

Table 230: 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 structure SHALL contain one or more Credential structures.

The authentication mechanisms are described and discussed in Section 8.

Object	Encoding
Authentication	Structure
Credential, MAY be repeated	Structure, see 2.1.2

Table 231: 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, the server SHALL process the request synchronously.

Object	Encoding
Asynchronous Indicator	Boolean

Table 232: 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 (see 6.7). A server-generated correlation value SHALL be specified in any subsequent Poll or Cancel operations that pertain to the original operation.

Object	Encoding
Asynchronous Correlation Value	Byte String

Table 233: 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 and 7.2) was set to Undo).
- *Operation Failed* – The requested operation failed.

Object	Encoding
Result Status	Enumeration, see 9.1.3.2.28

Table 234: 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:

- *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 could not be validated, or was not found.
- *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 REQUIRED additional 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.

- *Application Namespace Not Supported* – The particular Application Namespace is not supported, and the 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 9.1.3.2.29

Table 235: Result Reason in Response Batch Item

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.

Object	Encoding
Result Message	Text String

Table 236: 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 237: 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 9.1.3.2.30

Option	

Table 238: 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 239: 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 Section 7.2.

Object	Encoding
Batch Item	Structure

Table 240: 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 that SHALL contain the Vendor Identification, Criticality Indicator, and Vendor Extension fields. 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. The *Vendor Extension* structure SHALL contain vendor-specific extensions.

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

Table 241: Message Extension Structure in Batch Item

6.17 Attestation Capable Indicator

The *Attestation Capable Indicator* flag indicates whether the client is able to create an Attestation Credential object. It SHALL have Boolean value True if the client is able to create an Attestation Credential object, and the value False otherwise. If not present, the value False is assumed. If a client indicates that it is not able to create an Attestation Credential Object, and the client has issued an operation that requires attestation such as Get, then the server SHALL respond to the request with a failure.

Object	**Encoding**
Attestation Capable Indicator	Boolean

2069 *Table 242: Attestation Capable Indicator in Message Request Header*

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 245	Yes
Batch Item	Structure, see Table 246	Yes, MAY be repeated

Table 243: Request Message Structure

Object	Encoding	REQUIRED
Response Message	Structure	
Response Header	Structure, see Table 247	Yes
Batch Item	Structure, see Table 248	Yes, MAY be repeated

Table 244: Response Message Structure

7.2 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.

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	No	See 6.7
Attestation Capable Indicator	No	See 6.17
Attestation Type	No, MAY be repeated	See 9.1.3.2.36
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

2079 *Table 245: Request Header Structure*

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

2080 *Table 246: Request Batch Item Structure*

Response Header		
Object	**REQUIRED in Message**	**Comment**
Response Header	Yes	Structure
Protocol Version	Yes	See 6.1
Time Stamp	Yes	See 6.5
Nonce	No	See 2.1.14
Attestation Type	No, MAY be repeated	REQUIRED in *Attestation Required* error message if client set Attestation Capable Indicator to True in the request, see 9.1.3.2.36
Batch Count	Yes	See 6.14

2081 *Table 247: Response Header Structure*

Response Batch Item		
Object	**REQUIRED in Message**	**Comment**
Batch Item	Yes	Structure, see 6.15
Operation	Yes, if specified in Request Batch Item	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*	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	No	REQUIRED 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

2082 *Table 248: Response Batch Item Structure*

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

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 249: 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 250: 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 (deprecated as of version 1.1)
Certificate Issuer	420015 (deprecated as of version 1.1)
Certificate Issuer Alternative Name	420016 (deprecated as of version 1.1)
Certificate Issuer Distinguished Name	420017 (deprecated as of version 1.1)
Certificate Request	420018
Certificate Request Type	420019

Tag	
Object	**Tag Value**
Certificate Subject	`42001A` (deprecated as of version 1.1)
Certificate Subject Alternative Name	`42001B` (deprecated as of version 1.1)
Certificate Subject Distinguished Name	`42001C` (deprecated as of version 1.1)
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` (deprecated as of version 1.1)

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**
Key Role Type	`420083`
Salt	`420084`
Secret Data	`420085`
Secret Data Type	`420086`
Serial Number	`420087` (deprecated as of version 1.1)
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 Count	`420096`
Usage Limits Total	`420097`
Usage Limits Unit	`420098`
Username	`420099`
Validity Date	`42009A`
Validity Indicator	`42009B`
Vendor Extension	`42009C`
Vendor Identification	`42009D`
Wrapping Method	`42009E`
X	`42009F`
Y	`4200A0`
Password	`4200A1`
Device Identifier	`4200A2`
Encoding Option	`4200A3`
Extension Information	`4200A4`
Extension Name	`4200A5`
Extension Tag	`4200A6`

Tag	
Object	**Tag Value**
Extension Type	4200A7
Fresh	4200A8
Machine Identifier	4200A9
Media Identifier	4200AA
Network Identifier	4200AB
Object Group Member	4200AC
Certificate Length	4200AD
Digital Signature Algorithm	4200AE
Certificate Serial Number	4200AF
Device Serial Number	4200B0
Issuer Alternative Name	4200B1
Issuer Distinguished Name	4200B2
Subject Alternative Name	4200B3
Subject Distinguished Name	4200B4
X.509 Certificate Identifier	4200B5
X.509 Certificate Issuer	4200B6
X.509 Certificate Subject	4200B7
Key Value Location	4200B8
Key Value Location Value	4200B9
Key Value Location Type	4200BA
Key Value Present	4200BB
Original Creation Date	4200BC
PGP Key	4200BD
PGP Key Version	4200BE
Alternative Name	4200BF
Alternative Name Value	4200C0
Alternative Name Type	4200C1
Data	4200C2
Signature Data	4200C3
Data Length	4200C4
Random IV	4200C5
MAC Data	4200C6
Attestation Type	4200C7
Nonce	4200C8
Nonce ID	4200C9
Nonce Value	4200CA

Tag	
Object	**Tag Value**
Attestation Measurement	4200CB
Attestation Assertion	4200CC
IV Length	4200CD
Tag Length	4200CE
Fixed Field Length	4200CF
Counter Length	4200D0
Initial Counter Value	4200D1
Invocation Field Length	4200D2
Attestation Capable Indicator	4200D3
(Reserved)	4200D4 – 42FFFF
(Unused)	430000 – 53FFFF
Extensions	540000 – 54FFFF
(Unused)	550000 - FFFFFF

Table 251: Tag Values

9.1.3.2 Enumerations

The following tables define the values for enumerated lists. Values not listed (outside the range 80000000 to 8FFFFFFF) are reserved for future KMIP versions.

9.1.3.2.1 Credential Type Enumeration

Credential Type	
Name	**Value**
Username and Password	00000001
Device	00000002
Attestation	00000003
Extensions	8XXXXXXX

Table 252: 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 253: 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 254: 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 255: Wrapping Method Enumeration

9.1.3.2.5 Recommended Curve Enumeration

Recommended curves are defined in **[FIPS186-4] [SEC2] [X9.62] [ECC-Brainpool][RFC5639]**,

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`
SECP112R1	`00000010`
SECP112R2	`00000011`
SECP128R1	`00000012`
SECP128R2	`00000013`
SECP160K1	`00000014`
SECP160R1	`00000015`
SECP160R2	`00000016`
SECP192K1	`00000017`

SECP224K1	00000018
SECP256K1	00000019
SECT113R1	0000001A
SECT113R2	0000001B
SECT131R1	0000001C
SECT131R2	0000001D
SECT163R1	0000001E
SECT193R1	0000001F
SECT193R2	00000020
SECT239K1	00000021
ANSIX9P192V2	00000022
ANSIX9P192V3	00000023
ANSIX9P239V1	00000024
ANSIX9P239V2	00000025
ANSIX9P239V3	00000026
ANSIX9C2PNB163V1	00000027
ANSIX9C2PNB163V2	00000028
ANSIX9C2PNB163V3	00000029
ANSIX9C2PNB176V1	0000002A
ANSIX9C2TNB191V1	0000002B
ANSIX9C2TNB191V2	0000002C
ANSIX9C2TNB191V3	0000002D
ANSIX9C2PNB208W1	0000002E
ANSIX9C2TNB239V1	0000002F
ANSIX9C2TNB239V2	00000030
ANSIX9C2TNB239V3	00000031
ANSIX9C2PNB272W1	00000032
ANSIX9C2PNB304W1	00000033
ANSIX9C2TNB359V1	00000034
ANSIX9C2PNB368W1	00000035
ANSIX9C2TNB431R1	00000036
BRAINPOOLP160R1	00000037
BRAINPOOLP160T1	00000038
BRAINPOOLP192R1	00000039
BRAINPOOLP192T1	0000003A
BRAINPOOLP224R1	0000003B
BRAINPOOLP224T1	0000003C
BRAINPOOLP256R1	0000003D

BRAINPOOLP256T1	`0000003E`
BRAINPOOLP320R1	`0000003F`
BRAINPOOLP320T1	`00000040`
BRAINPOOLP384R1	`00000041`
BRAINPOOLP384T1	`00000042`
BRAINPOOLP512R1	`00000043`
BRAINPOOLP512T1	`00000044`
Extensions	`8XXXXXXX`

Table 256: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV

9.1.3.2.6 Certificate Type Enumeration

The PGP certificate type is deprecated as of version 1.2 of this specification and MAY be removed from subsequent versions of the specification.

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

Table 257: Certificate Type Enumeration

9.1.3.2.7 Digital Signature Algorithm Enumeration

Digital Signature Algorithm	
Name	**Value**
MD2 with RSA Encryption (PKCS#1 v1.5)	`00000001`
MD5 with RSA Encryption (PKCS#1 v1.5)	`00000002`
SHA-1 with RSA Encryption (PKCS#1 v1.5)	`00000003`
SHA-224 with RSA Encryption (PKCS#1 v1.5)	`00000004`
SHA-256 with RSA Encryption (PKCS#1 v1.5)	`00000005`
SHA-384 with RSA Encryption (PKCS#1 v1.5)	`00000006`
SHA-512 with RSA Encryption (PKCS#1 v1.5)	`00000007`
RSASSA-PSS (PKCS#1 v2.1)	`00000008`
DSA with SHA-1	`00000009`
DSA with SHA224	`0000000A`
DSA with SHA256	`0000000B`
ECDSA with SHA-1	`0000000C`
ECDSA with SHA224	`0000000D`
ECDSA with SHA256	`0000000E`
ECDSA with SHA384	`0000000F`
ECDSA with SHA512	`00000010`
Extensions	`8XXXXXXX`

Table 258: Digital Signature Algorithm Enumeration

9.1.3.2.8 Split Key Method Enumeration

Split Key Method	
Name	**Value**
XOR	`00000001`
Polynomial Sharing GF (2^{16})	`00000002`
Polynomial Sharing Prime Field	`00000003`
Polynomial Sharing GF (2^{8})	`00000004`
Extensions	`8XXXXXXX`

Table 259: Split Key Method Enumeration

9.1.3.2.9 Secret Data Type Enumeration

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

Table 260: Secret Data Type Enumeration

9.1.3.2.10 Opaque Data Type Enumeration

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

Table 261: Opaque Data Type Enumeration

9.1.3.2.11 Name Type Enumeration

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

Table 262: Name Type Enumeration

9.1.3.2.12 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
PGP Key	00000009
Extensions	8XXXXXXX

Table 263: Object Type Enumeration

2214
9.1.3.2.13 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`
Blowfish	`00000010`
Camellia	`00000011`
CAST5	`00000012`
IDEA	`00000013`
MARS	`00000014`
RC2	`00000015`
RC4	`00000016`
RC5	`00000017`
SKIPJACK	`00000018`
Twofish	`00000019`
EC	`0000001A`
Extensions	`8XXXXXXX`

2215 *Table 264: Cryptographic Algorithm Enumeration*

2216 9.1.3.2.14 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`

2217 *Table 265: Block Cipher Mode Enumeration*

2218 9.1.3.2.15 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`

2219 *Table 266: Padding Method Enumeration*

9.1.3.2.16 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`
RIPEMD-160	`00000009`
Tiger	`0000000A`
Whirlpool	`0000000B`
SHA-512/224	`0000000C`
SHA-512/256	`0000000D`
Extensions	`8XXXXXXX`

Table 267: Hashing Algorithm Enumeration

9.1.3.2.17 Key Role Type Enumeration

Key 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 268: Key Role Type Enumeration

Note that while the set and definitions of key role types are chosen to match **[X9 TR-31]** there is no necessity to match binary representations.

9.1.3.2.18 State Enumeration

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

Extensions	8XXXXXXX

2227 *Table 269: State Enumeration*

2228 9.1.3.2.19 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

2229 *Table 270: Revocation Reason Code Enumeration*

2230 9.1.3.2.20 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
Parent Link	00000108
Child Link	00000109
Previous Link	0000010A
Next Link	0000010B
Extensions	8XXXXXXX

2231 *Table 271: Link Type Enumeration*

9.1.3.2.21 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 272: Derivation Method Enumeration

9.1.3.2.22 Certificate Request Type Enumeration

The PGP certificate request type is deprecated as of version 1.1 of this specification and MAY be removed from subsequent versions of the specification.

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

Table 273: Certificate Request Type Enumeration

9.1.3.2.23 Validity Indicator Enumeration

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

Table 274: Validity Indicator Enumeration

9.1.3.2.24 Query Function Enumeration

Query Function	
Name	**Value**
Query Operations	`00000001`
Query Objects	`00000002`
Query Server Information	`00000003`
Query Application Namespaces	`00000004`
Query Extension List	`00000005`
Query Extension Map	`00000006`
Query Attestation Types	`00000007`
Extensions	`8XXXXXXX`

Table 275: Query Function Enumeration

9.1.3.2.25 Cancellation Result Enumeration

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

Table 276: Cancellation Result Enumeration

9.1.3.2.26 Put Function Enumeration

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

Table 277: Put Function Enumeration

2246
9.1.3.2.27 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
Re-key Key Pair	0000001D
Discover Versions	0000001E
Encrypt	0000001F
Decrypt	00000020
Sign	00000021
Signature Verify	00000022
MAC	00000023

MAC Verify	00000024
RNG Retrieve	00000025
RNG Seed	00000026
Hash	00000027
Create Split Key	00000028
Join Split Key	00000029
Extensions	8XXXXXXX

Table 278: Operation Enumeration

9.1.3.2.28 Result Status Enumeration

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

Table 279: Result Status Enumeration

2250

9.1.3.2.29 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 Not Supported	00000010
Key Compression Type Not Supported	00000011
Encoding Option Error	00000012
Key Value Not Present	00000013
Attestation Required	00000014
Attestation Failed	00000015
General Failure	00000100
Extensions	8XXXXXXX

2251 *Table 280: Result Reason Enumeration*

2252

9.1.3.2.30 Batch Error Continuation Option Enumeration

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

Table 281: Batch Error Continuation Option Enumeration

9.1.3.2.31 Usage Limits Unit Enumeration

Usage Limits Unit	
Name	**Value**
Byte	`00000001`
Object	`00000002`
Extensions	`8XXXXXXX`

Table 282: Usage Limits Unit Enumeration

9.1.3.2.32 Encoding Option Enumeration

Encoding Option	
Name	**Value**
No Encoding	`00000001`
TTLV Encoding	`00000002`
Extensions	`8XXXXXXX`

Table 283: Encoding Option Enumeration

9.1.3.2.33 Object Group Member Enumeration

Object Group Member Option	
Name	**Value**
Group Member Fresh	`00000001`
Group Member Default	`00000002`
Extensions	`8XXXXXXX`

Table 284: Object Group Member Enumeration

9.1.3.2.34 Alternative Name Type Enumeration

Alternative Name Type	
Name	**Value**
Uninterpreted Text String	`00000001`
URI	`00000002`
Object Serial Number	`00000003`
Email Address	`00000004`
DNS Name	`00000005`
X.500 Distinguished Name	`00000006`
IP Address	`00000007`
Extensions	`8XXXXXXX`

Table 285: Alternative Name Type Enumeration

9.1.3.2.35 Key Value Location Type Enumeration

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

Table 286: Key Value Location Type Enumeration

9.1.3.2.36 Attestation Type Enumeration

Attestation Type	
Name	**Value**
TPM Quote	00000001
TCG Integrity Report	00000002
SAML Assertion	00000003
Extensions	8XXXXXXX

Table 287: Attestation Type 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 288: 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 289: Storage Status Mask

10 Transport

KMIP Servers and Clients SHALL establish and maintain channel confidentiality and integrity, and provide assurance of authenticity for KMIP messaging 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
Maximum Response Size has been exceeded	Result Status is Operation Failed	Response Too Large
Server does not support operation	Result Status is Operation Failed	Operation Not Supported
The Criticality Indicator in a Message Extension structure is set to True, but the server does not understand the extension	Result Status is Operation Failed	Feature Not Supported
Message cannot be parsed	Response message containing a header and a Batch Item without Operation, but with the Result Status field set to Operation Failed	Invalid Message
Operation requires attestation data which was not provided by the client, and the client has set the Attestation	Result Status is Operation Failed	Attestation Required

Capable indicator to True		
Operation requires attestation data which was not provided by the client, and the client has set the Attestation Capable indicator to False	Result Status is Operation Failed	Permission Denied
Operation requires attestation data and the attestation data provided by the client does not validate	Result Status is Operation Failed	Attestation Failed

Table 290: 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

Table 291: 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

2285 *Table 292: Create Key Pair Errors*

2286 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 Template object containing a Name attribute with the Template structure	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
Encoding Option not permitted when Key Wrapping Specification contains attribute names	Operation Failed	Encoding Option Error

2287 *Table 293: Register Errors*

2288 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
An offset cannot be used to specify new Process Start, Protect Stop and/or Deactivation Date attribute values since no Activation Date has been specified for the existing key	Operation Failed	Illegal Operation
The Key Value is not present on the server	Operation Failed	Key Value Not Present

2289 *Table 294: Re-key Errors*

2290 11.6 Re-key Key Pair

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 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

An offset cannot be used to specify new Process Start, Protect Stop and/or Deactivation Date attribute values since no Activation Date has been specified for the existing key	Operation Failed	Illegal Operation
The Key Value is not present on the server	Operation Failed	Key Value Not Present

Table 295: Re-key Key Pair Errors

11.7 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
The specified length exceeds the output of the derivation method or other cryptographic error during derivation.	Operation Failed	Cryptographic Failure
The Key Value is not present on the server	Operation Failed	Key Value Not Present

Table 296: Derive Key Errors-

11.8 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
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 297: Certify Errors

11.9 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
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 298: Re-certify Errors

11.10 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 299: Locate Errors

11.11 Check

Error Definition	Result Status	Result Reason
Object does not exist	Operation Failed	Item Not Found
Object is archived	Operation Failed	Object Archived
Check cannot be performed on this object	Operation Failed	Illegal Operation
The client is not allowed to use the object according to the specified attributes	Operation Failed	Permission Denied

Table 300: Check Errors

11.12 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
Object exists but cannot be provided in the desired Encoding Option	Operation Failed	Encoding Option Error
Encoding Option not permitted when	Operation Failed	Encoding Option Error

Key Wrapping Specification contains attribute names		

Table 301: Get Errors

11.13 Get Attributes

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
The same Attribute Name is present more than once	Operation Failed	Invalid Message
Object is archived	Operation Failed	Object Archived

Table 302: Get Attributes Errors

11.14 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 303: Get Attribute List Errors

11.15 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 304: Add Attribute Errors

11.16 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
No matching attribute 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 from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived

Table 305: Modify Attribute Errors

11.17 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
No matching attribute instance exists	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 306: Delete Attribute Errors

11.18 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 307: Obtain Lease Errors

11.19 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
No Usage Limits Count is specified	Operation Failed	Invalid Message
Object is archived	Operation Failed	Object Archived
The server was not able to grant the requested amount of usage allocation	Operation Failed	Permission Denied

Table 308: Get Usage Allocation Errors

11.20 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 309: Activate Errors

11.21 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 310: Revoke Errors

11.22 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 311: Destroy Errors

11.23 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 312: Archive Errors

11.24 Recover

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

Table 313: Recover Errors

11.25 Validate

Error Definition	Result Status	Result Reason
The combination of Certificate Objects and Unique Identifiers does not specify	Operation Failed	Invalid Message

a certificate list		
One or more of the objects is archived	Operation Failed	Object Archived

Table 314: Validate Errors

11.26 Query

N/A

11.27 Cancel

N/A

11.28 Poll

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

Table 315: Poll Errors

11.29 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

Table 316: Batch Items Errors

2342 11.30 Create Split Key Errors

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
Split Key Method not supported	Operation Failed	Invalid Field
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found

2343 *Table 317: Create Split Key Errors*

2344 11.31 Join Split Key Errors

2345

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
Number of Unique Identifiers given in request is less than Split Key Threshold	Operation Failed	Cryptographic Failure?
Split Key Method not supported	Operation Failed	Invalid Field
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
One or more of the objects is archived	Operation Failed	Object Archived

Table 318: Join Split Key Errors

12 KMIP Server and Client Implementation Conformance

12.1 KMIP Server Implementation Conformance

An implementation is a conforming KMIP Server if the implementation meets the conditions specified in one or more server profiles specified in **[KMIP-Prof]**.

A KMIP server implementation SHALL be a conforming KMIP Server.

If a KMIP server implementation claims support for a particular server profile, then the implementation SHALL conform to all normative statements within the clauses specified for that profile and for any subclauses to each of those clauses.

12.2 KMIP Client Implementation Conformance

An implementation is a conforming KMIP Client if the implementation meets the conditions specified in one or more client profiles specified in **[KMIP-Prof]**.

A KMIP client implementation SHALL be a conforming KMIP Client.

If a KMIP client implementation claims support for a particular client profile, then the implementation SHALL conform to all normative statements within the clauses specified for that profile and for any subclauses to each of those clauses.

Appendix A. Acknowledgments

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

Participants:

Hal Aldridge, Sypris Electronics
Mike Allen, Symantec
Gordon Arnold, IBM
Todd Arnold, IBM
Richard Austin, Hewlett-Packard
Lars Bagnert, PrimeKey
Elaine Barker, NIST
Peter Bartok, Venafi, Inc.
Tom Benjamin, IBM
Anthony Berglas, Cryptsoft
Mathias Björkqvist, IBM
Kevin Bocket, Venafi
Anne Bolgert, IBM
Alan Brown, Thales e-Security
Tim Bruce, CA Technologies
Chris Burchett, Credant Technologies, Inc.
Kelley Burgin, National Security Agency
Robert Burns, Thales e-Security
Chuck Castleton, Venafi
Kenli Chong, QuintessenceLabs
John Clark, Hewlett-Packard
Tom Clifford, Symantec Corp.
Tony Cox, Cryptsoft
Russell Dietz, SafeNet, Inc
Graydon Dodson, Lexmark International Inc.
Vinod Duggirala, EMC Corporation
Chris Dunn, SafeNet, Inc.
Michael Duren, Sypris Electronics
James Dzierzanowski, American Express CCoE
Faisal Faruqui, Thales e-Security
Stan Feather, Hewlett-Packard
David Finkelstein, Symantec Corp.
James Fitzgerald, SafeNet, Inc.
Indra Fitzgerald, Hewlett-Packard
Judith Furlong, EMC Corporation
Susan Gleeson, Oracle
Robert Griffin, EMC Corporation
Paul Grojean, Individual
Robert Haas, IBM
Thomas Hardjono, M.I.T.
ChengDong He, Huawei Technologies Co., Ltd.
Steve He, Vormetric
Kurt Heberlein, Hewlett-Packard
Larry Hofer, Emulex Corporation
Maryann Hondo, IBM
Walt Hubis, NetApp
Tim Hudson, Cryptsoft
Jonas Iggbom, Venafi, Inc.
Sitaram Inguva, American Express CCoE

Jay Jacobs, Target Corporation
Glen Jaquette, IBM
Mahadev Karadiguddi, NetApp
Greg Kazmierczak, Wave Systems Corp.
Marc Kenig, SafeNet Inc.
Mark Knight, Thales e-Security
Kathy Kriese, Symantec Corporation
Mark Lambiase, SecureAuth
John Leiseboer, Quintenssence Labs
Hal Lockhart, Oracle Corporation
Robert Lockhart, Thales e-Security
Anne Luk, Cryptsoft
Sairam Manidi, Freescale
Luther Martin, Voltage Security
Neil McEvoy, iFOSSF
Marina Milshtein, Individual
Dale Moberg, Axway Software
Jishnu Mukeri, Hewlett-Packard
Bryan Olson, Hewlett-Packard
John Peck, IBM
Rob Philpott, EMC Corporation
Denis Pochuev, SafeNet, Inc.
Reid Poole, Venafi, Inc.
Ajai Puri, SafeNet, Inc.
Saravanan Ramalingam, Thales e-Security
Peter Reed, SafeNet, Inc.
Bruce Rich, IBM
Christina Richards, American Express CCoE
Warren Robbins, Dell
Peter Robinson, EMC Corporation
Scott Rotondo, Oracle
Saikat Saha, Oracle
Anil Saldhana, Red Hat
Subhash Sankuratripati, NetApp
Boris Schumperli, Cryptomathic
Greg Singh, QuintessenceLabs
David Smith, Venafi, Inc.
Brian Spector, Certivox
Terence Spies, Voltage Security
Deborah Steckroth, RouteOne LLC
Michael Stevens, QuintessenceLabs
Marcus Streets, Thales e-Security
Satish Sundar, IBM
Kiran Thota, VMware
Somanchi Trinath, Freescale Semiconductor, Inc.
Nathan Turajski, Thales e-Security
Sean Turner, IECA, Inc.
Paul Turner, Venafi, Inc.
Rod Wideman, Quantum Corporation
Steven Wierenga, Hewlett-Packard
Jin Wong, QuintessenceLabs
Sameer Yami, Thales e-Security
Peter Yee, EMC Corporation
Krishna Yellepeddy, IBM
Catherine Ying, SafeNet, Inc.
Tatu Ylonen, SSH Communications Security (Tectia Corp)
Michael Yoder, Vormetric. Inc.

Magda Zdunkiewicz, Cryptsoft
Peter Zelechoski, Election Systems & Software

Appendix B. Attribute Cross-Reference

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

Attribute Name	Managed Object								
	Certificate	Symmetric Key	Public Key	Private Key	Split Key	Template	Secret Data	Opaque Object	PGP Key
Unique Identifier	x	x	x	x	x	x	x	x	x
Name	x	x	x	x	x	x	x	x	x
Object Type	x	x	x	x	x	x	x	x	x
Cryptographic Algorithm	x	x	x	x	x	x			x
Cryptographic Domain Parameters			x	x		x			
Cryptographic Length	x	x	x	x	x	x			x
Cryptographic Parameters	x	x	x	x	x	x			x
Certificate Type	x								x
Certificate Identifier	x								x
Certificate Issuer	x								x
Certificate Length	x								x
Certificate Subject	x								x
Digital Signature Algorithm	x								x
Digest	x	x	x	x	x		x		x
Operation Policy Name	x	x	x	x	x	x	x	x	x
Cryptographic Usage Mask	x	x	x	x	x	x	x		x
Lease Time	x	x	x	x	x		x	x	x
Usage Limits		x	x	x	x	x			
State	x	x	x	x	x		x		x
Initial Date	x	x	x	x	x	x	x	x	x
Activation Date	x	x	x	x	x	x	x		x
Process Start Date		x			x	x			
Protect Stop Date		x			x	x			
Deactivation Date	x	x	x	x	x	x	x	x	x
Destroy Date	x	x	x	x	x		x	x	x
Compromise Occurrence Date	x	x	x	x	x		x	x	x

Attribute Name	**Managed Object**								
	Certificate	**Symmetric Key**	**Public Key**	**Private Key**	**Split Key**	**Template**	**Secret Data**	**Opaque Object**	**PGP Key**
Compromise Date	x	x	x	x	x		x	x	x
Revocation Reason	x	x	x	x	x		x	x	x
Archive Date	x	x	x	x	x	x	x	x	x
Object Group	x	x	x	x	x	x	x	x	x
Fresh	x	x	x	x	x				x
Link	x	x	x	x	x		x		x
Application Specific Information	x	x	x	x	x	x	x	x	x
Contact Information	x	x	x	x	x	x	x	x	x
Last Change Date	x	x	x	x	x	x	x	x	x
Custom Attribute	x	x	x	x	x	x	x	x	x
Alternative Name	x	x	x	x	x	x	x	x	x
Key Value Present		x		x	x		x		
Key Value Location		x		x	x		x		
Original Creation Date	x	x	x	x	x	x	x	x	x

Table 319: Attribute Cross-reference

Appendix C. Tag Cross-Reference

This table is not normative.

Object	Defined	Type	Notes
Activation Date	3.24	Date-Time	
Application Data	3.36	Text String	
Application Namespace	3.36	Text String	
Application Specific Information	3.36	Structure	
Archive Date	3.32	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, 9.1.3.2.30	Enumeration	
Batch Item	6.15	Structure	
Batch Order Option	6.12	Boolean	
Block Cipher Mode	3.6, 9.1.3.2.14	Enumeration	
Cancellation Result	4.27, 9.1.3.2.25	Enumeration	
Certificate	2.2.1	Structure	
Certificate Identifier	3.13	Structure	deprecated as of version 1.1
Certificate Issuer	3.13	Structure	deprecated as of version 1.1
Certificate Issuer Alternative Name	3.15	Text String	deprecated as of version 1.1
Certificate Issuer Distinguished Name	3.15	Text String	deprecated as of version 1.1
Certificate Length	3.9	Integer	
Certificate Request	4.7, 4.8	Byte String	
Certificate Request Type	4.7, 4.8, 9.1.3.2.22	Enumeration	
Certificate Serial Number	3.9	Byte String	
Certificate Subject	3.14	Structure	deprecated as of version 1.1
Certificate Subject Alternative Name	3.14	Text String	deprecated as of version 1.1
Certificate Subject Distinguished Name	3.14	Text String	deprecated as of version 1.1

Object	Defined	Type	Notes
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	3.29	Date-Time	
Compromise Date	3.30	Date-Time	
Contact Information	3.37	Text String	
Credential	2.1.2	Structure	
Credential Type	2.1.2, 9.1.3.2.1	Enumeration	
Credential Value	2.1.2	*	type varies
Criticality Indicator	6.16	Boolean	
CRT Coefficient	2.1.7	Big Integer	
Cryptographic Algorithm	3.4, 9.1.3.2.13	Enumeration	
Cryptographic Length	3.5	Integer	
Cryptographic Parameters	3.6	Structure	
Cryptographic Usage Mask	3.19, 9.1.3.3.1	Integer	Bit mask
Custom Attribute	3.39	*	type varies
D	2.1.7	Big Integer	
Deactivation Date	3.27	Date-Time	
Derivation Data	4.6	Byte String	
Derivation Method	4.6, 9.1.3.2.21	Enumeration	
Derivation Parameters	4.6	Structure	
Destroy Date	3.28	Date-Time	
Device Identifier	2.1.2	Text String	
Device Serial Number	2.1.2	Text String	
Digest	3.17	Structure	
Digest Value	3.17	Byte String	
Digital Signature Algorithm	3.16	Enumeration	
Encoding Option	2.1.5, 2.1.6, 9.1.3.2.32	Enumeration	
Encryption Key Information	2.1.5	Structure	
Extension Information	2.1.9	Structure	
Extension Name	2.1.9	Text String	
Extension Tag	2.1.9	Integer	
Extension Type	2.1.9	Integer	
Extensions	9.1.3		
Fresh	3.34	Boolean	
G	2.1.7	Big Integer	
Hashing Algorithm	3.6, 3.17, 9.1.3.2.16	Enumeration	
Initial Date	3.23	Date-Time	

Object	**Defined**	**Type**	**Notes**
Initialization Vector	4.6	Byte String	
Issuer	3.13	Text String	deprecated as of version 1.1
Issuer Alternative Name	3.12	Byte String	
Issuer Distinguished Name	3.12	Byte String	
Iteration Count	4.6	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 Role Type	3.6, 9.1.3.2.17	Enumeration	
Key Value	2.1.4	Byte String / Structure	
Key Wrapping Data	2.1.5	Structure	
Key Wrapping Specification	2.1.6	Structure	
Last Change Date	3.38	Date-Time	
Lease Time	3.20	Interval	
Link	3.35	Structure	
Link Type	3.35, 9.1.3.2.20	Enumeration	
Linked Object Identifier	3.35	Text String	
MAC/Signature	2.1.5	Byte String	
MAC/Signature Key Information	2.1.5	Text String	
Machine Identifier	2.1.2	Text String	
Maximum Items	4.9	Integer	
Maximum Response Size	6.3	Integer	
Media Identifier	2.1.2	Text String	
Message Extension	6.16	Structure	
Modulus	2.1.7	Big Integer	
Name	3.2	Structure	
Name Type	3.2, 9.1.3.2.11	Enumeration	
Name Value	3.2	Text String	
Network Identifier	2.1.2	Text String	
Object Group	3.33	Text String	
Object Group Member	4.9	Enumeration	
Object Type	3.3, 9.1.3.2.12	Enumeration	

Object	**Defined**	**Type**	**Notes**
Offset	4.4, 4.8	Interval	
Opaque Data Type	2.2.8, 9.1.3.2.10	Enumeration	
Opaque Data Value	2.2.8	Byte String	
Opaque Object	2.2.8	Structure	
Operation	6.2, 9.1.3.2.27	Enumeration	
Operation Policy Name	3.18	Text String	
P	2.1.7	Big Integer	
Password	2.1.2	Text String	
Padding Method	3.6, 9.1.3.2.15	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	3.25	Date-Time	
Protect Stop Date	3.26	Date-Time	
Protocol Version	6.1	Structure	
Protocol Version Major	6.1	Integer	
Protocol Version Minor	6.1	Integer	
Public Exponent	2.1.7	Big Integer	
Public Key	2.2.3	Structure	
Public Key Template-Attribute	2.1.8	Structure	
Public Key Unique Identifier	4.2	Text String	
Put Function	5.2, 9.1.3.2.26	Enumeration	
Q	2.1.7	Big Integer	
Q String	2.1.7	Byte String	
Qlength	3.7	Integer	
Query Function	4.25, 9.1.3.2.24	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	Structure	
Request Message	7.1	Structure	
Request Payload	4, 5, 7.2	Structure	
Response Header	7.2	Structure	
Response Message	7.1	Structure	
Response Payload	4, 7.2	Structure	

Object	Defined	Type	Notes
Result Message	6.11	Text String	
Result Reason	6.10, 9.1.3.2.29	Enumeration	
Result Status	6.9, 9.1.3.2.28	Enumeration	
Revocation Message	3.31	Text String	
Revocation Reason	3.31	Structure	
Revocation Reason Code	3.31, 9.1.3.2.19	Enumeration	
Salt	4.6	Byte String	
Secret Data	2.2.7	Structure	
Secret Data Type	2.2.7, 9.1.3.2.9	Enumeration	
Serial Number	3.13	Text String	deprecated as of version 1.1
Server Information	4.25	Structure	contents vendor-specific
Split Key	2.2.5	Structure	
Split Key Method	2.2.5, 9.1.3.2.8	Enumeration	
Split Key Parts	2.2.5	Integer	
Split Key Threshold	2.2.5	Integer	
State	3.22, 9.1.3.2.18	Enumeration	
Storage Status Mask	4.9, 9.1.3.3.2	Integer	Bit mask
Subject Alternative Name	3.11	Byte String	
Subject Distinguished Name	3.11	Byte String	
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	
Unique Identifier	3.1	Text String	
Unique Batch Item ID	6.4	Byte String	
Username	2.1.2	Text String	
Usage Limits	3.21	Structure	
Usage Limits Count	3.21	Long Integer	
Usage Limits Total	3.21	Long Integer	
Usage Limits Unit	3.21	Enumeration	
Validity Date	4.24	Date-Time	
Validity Indicator	4.24, 9.1.3.2.23	Enumeration	
Vendor Extension	6.16	Structure	contents vendor-specific
Vendor Identification	4.25, 6.16	Text String	
Wrapping Method	2.1.5, 9.1.3.2.4	Enumeration	
X	2.1.7	Big Integer	

Object	Defined	Type	Notes
X.509 Certificate Identifier	3.9	Structure	
X.509 Certificate Issuer	3.12	Structure	
X.509 Certificate Subject	3.11	Structure	
Y	2.1.7	Big Integer	

Table 320: Tag Cross-reference

Appendix D. Operations 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.

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

Operation	Managed Objects								
	Certificate	Symmetric Key	Public Key	Private Key	Split Key	Template	Secret Data	Opaque Object	PGP Key
Poll	N/A	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	N/A
Put	Y	Y	Y	Y	Y	Y	Y	Y	Y
Discover Versions	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 321: Operation and Object Cross-reference

Appendix E. 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 **[FIPS197]**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 **[SP800-38C]**
CFB - Cipher Feedback specified in **[SP800-38A]**
CMAC - Cipher-based MAC specified in **[SP800-38B]**
CMC - Certificate Management Messages over CMS specified in **[RFC5272]**
CMP - Certificate Management Protocol specified in **[RFC4210]**
CPU - Central Processing Unit
CRL - Certificate Revocation List specified in **[RFC5280]**
CRMF - Certificate Request Message Format specified in **[RFC4211]**
CRT - Chinese Remainder Theorem
CTR - Counter specified in **[SP800-38A]**
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 **[X9.63][SP800-56A]**
ECDSA - Elliptic Curve Digital Signature Algorithm specified in **[X9.62]**
ECMQV - Elliptic Curve Menezes Qu Vanstone specified in **[X9.63][SP800-56A]**
FFC - Finite Field Cryptography
FIPS - Federal Information Processing Standard
GCM - Galois/Counter Mode specified in **[SP800-38D]**
GF - Galois field (or finite field)
HMAC - Keyed-Hash Message Authentication Code specified in **[FIPS198-1][RFC2104]**
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 **[RFC1319]**
MD4 - Message Digest 4 Algorithm specified in **[RFC1320]**
MD5 - Message Digest 5 Algorithm specified in **[RFC1321]**
NIST - National Institute of Standards and Technology
OAEP - Optimal Asymmetric Encryption Padding specified in **[PKCS#1]**
OFB - Output Feedback specified in **[SP800-38A]**
PBKDF2 - Password-Based Key Derivation Function 2 specified in **[RFC2898]**
PCBC - Propagating Cipher Block Chaining
PEM - Privacy Enhanced Mail specified in **[RFC1421]**
PGP - OpenPGP specified in **[RFC4880]**
PKCS - Public-Key Cryptography Standards
PKCS#1 - RSA Cryptography Specification Version 2.1 specified in **[RFC3447]**
PKCS#5 - Password-Based Cryptography Specification Version 2 specified in **[RFC2898]**
PKCS#8 - Private-Key Information Syntax Specification Version 1.2 specified in **[RFC5208]**
PKCS#10 - Certification Request Syntax Specification Version 1.7 specified in **[RFC2986]**
POSIX - Portable Operating System Interface
RFC - Request for Comments documents of IETF
RSA - Rivest, Shamir, Adelman (an algorithm)
SCEP - Simple Certificate Enrollment Protocol
SCVP - Server-based Certificate Validation 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	- Coordinated Universal Time
UTF-8	- Universal Transformation Format 8-bit specified in **[RFC3629]**
XKMS	- XML Key Management Specification
XML	- Extensible Markup Language
XTS	- XEX Tweakable Block Cipher with Ciphertext Stealing specified in **[SP800-38E]**
X.509	- Public Key Certificate specified in **[RFC5280]**
ZPK	- PIN Block Encryption Key specified in ANSI X9 TR-31

Appendix F. List of Figures and Tables

Appendix G. Revision History

Revision	Date	Editor	Changes Made
draft-01	2013-03-28	Kiran Thota	Ported to KMIP v1.2 Specification template
draft-02	2013-05-10	Kiran Thota	The following accepted proposals have been incorporated into the document: Meta Data Only PGP Key Alternative Key Cryptographic Services ECC Curves
draft-03	2013-05-12	Kiran Thota	The following proposals are also included – Attested operations Split Key
draft-04	2013-06-04	Kiran Thota Kelley Burgin	Addressed many comments for KMIP v1.1 Spec. Additions to Split Key
draft-05	2013-06-10	Kiran Thota	Incorporated feedback and updated Participants list
draft-06	2013-06-17	Kiran Thota	Incorporated feedback from Tim Hudson
draft-07	2013-08-08	Kiran Thota Kelley Burgin	Fixed typos, references. Added tags and definitions.
draft-08	2013-08-21	Kelley Burgin Kiran Thota	Fixed keywords. Editorial changes.
csd-01	2013-09-12	Kiran Thota	Updated references and participants list. Added enumerations for SHA512/224 and SHA512/256
csd-01 (revised)	2013-10-31	Kiran Thota	Fixed the tag table entries for nonce et al as Tim pointed out.
csd-01review1	2014-06-03	Kiran Thota	Incorporated feedback from public review
csd-01review3	2014-06-16	Kiran Thota	Added references. Made changes to Asynchronous Indicator and Attestation Capability Indicator. Few other editorial changes – fix tags, names etc.