



# Key Management Interoperability Protocol Specification Version 1.0

Committee Draft 06 / Public Review 01

05 November 2009

## Specification URI

### This Version:

<http://docs.oasis-open.org/kmip/spec/v1.0/cd06/kmip-spec-1.0-cd-06.html>  
<http://docs.oasis-open.org/kmip/spec/v1.0/cd06/kmip-spec-1.0-cd-06.doc> (Authoritative)  
<http://docs.oasis-open.org/kmip/spec/v1.0/cd06/kmip-spec-1.0-cd-06.pdf>

Previous Version:

N/A

### Latest Version:

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

### Technical Committee:

[OASIS Key Management Interoperability Protocol \(KMIP\) TC](#)

### Chair(s):

Robert Griffin, EMC Corporation <[robert.griffin@rsa.com](mailto:robert.griffin@rsa.com)>  
Subhash Sankuratripati, NetApp <[Subhash.Sankuratripati@netapp.com](mailto:Subhash.Sankuratripati@netapp.com)>

### Editor(s):

Robert Haas, IBM <[rha@zurich.ibm.com](mailto:rha@zurich.ibm.com)>  
Indra Fitzgerald, HP <[indra.fitzgerald@hp.com](mailto:indra.fitzgerald@hp.com)>

### Related work:

This specification replaces or supersedes:

- None

This specification is related to:

- [Key Management Interoperability Protocol Profiles Version 1.0](#)
- [Key Management Interoperability Protocol Use Cases Version 1.0](#)
- [Key Management Interoperability Protocol Usage Guide Version 1.0](#)

### Declared XML Namespace(s):

None

### Abstract:

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

### Status:

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

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

“Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/kmip/>.

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

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

---

## Notices

Copyright © OASIS® 2009. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", "KMIP" are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

---

# Table of Contents

1	Introduction .....	8
1.1	Terminology .....	8
1.2	Normative References .....	11
1.3	Non-normative References .....	13
2	Objects .....	14
2.1	Base Objects .....	14
2.1.1	Attribute .....	14
2.1.2	Credential .....	15
2.1.3	Key Block .....	15
2.1.4	Key Value .....	16
2.1.5	Key Wrapping Data .....	17
2.1.6	Key Wrapping Specification .....	18
2.1.7	Transparent Key Structures .....	19
2.1.8	Template-Attribute Structures .....	23
2.2	Managed Objects .....	23
2.2.1	Certificate .....	24
2.2.2	Symmetric Key .....	24
2.2.3	Public Key .....	24
2.2.4	Private Key .....	24
2.2.5	Split Key .....	24
2.2.6	Template .....	26
2.2.7	Secret Data .....	27
2.2.8	Opaque Object .....	27
3	Attributes .....	28
3.1	Unique Identifier .....	29
3.2	Name .....	29
3.3	Object Type .....	30
3.4	Cryptographic Algorithm .....	30
3.5	Cryptographic Length .....	31
3.6	Cryptographic Parameters .....	31
3.7	Cryptographic Domain Parameters .....	33
3.8	Certificate Type .....	34
3.9	Certificate Identifier .....	34
3.10	Certificate Subject .....	35
3.11	Certificate Issuer .....	36
3.12	Digest .....	36
3.13	Operation Policy Name .....	37
3.13.1	Operations outside of operation policy control .....	38
3.13.2	Default Operation Policy .....	38
3.14	Cryptographic Usage Mask .....	40
3.15	Lease Time .....	42
3.16	Usage Limits .....	42
3.17	State .....	44

3.18	Initial Date .....	46
3.19	Activation Date .....	46
3.20	Process Start Date .....	47
3.21	Protect Stop Date .....	47
3.22	Deactivation Date .....	48
3.23	Destroy Date .....	49
3.24	Compromise Occurrence Date .....	49
3.25	Compromise Date .....	49
3.26	Revocation Reason .....	50
3.27	Archive Date .....	51
3.28	Object Group .....	51
3.29	Link .....	51
3.30	Application Specific Information .....	53
3.31	Contact Information .....	53
3.32	Last Change Date .....	54
3.33	Custom Attribute .....	54
4	Client-to-Server Operations .....	56
4.1	Create .....	56
4.2	Create Key Pair .....	57
4.3	Register .....	59
4.4	Re-key .....	60
4.5	Derive Key .....	62
4.6	Certify .....	65
4.7	Re-certify .....	65
4.8	Locate .....	67
4.9	Check .....	69
4.10	Get .....	71
4.11	Get Attributes .....	71
4.12	Get Attribute List .....	72
4.13	Add Attribute .....	72
4.14	Modify Attribute .....	73
4.15	Delete Attribute .....	73
4.16	Obtain Lease .....	74
4.17	Get Usage Allocation .....	75
4.18	Activate .....	76
4.19	Revoke .....	76
4.20	Destroy .....	77
4.21	Archive .....	77
4.22	Recover .....	77
4.23	Validate .....	78
4.24	Query .....	78
4.25	Cancel .....	80
4.26	Poll .....	81
5	Server-to-Client Operations .....	82
5.1	Notify .....	82

5.2	Put.....	82
6	Message Contents.....	84
6.1	Protocol Version .....	84
6.2	Operation .....	84
6.3	Maximum Response Size .....	84
6.4	Unique Batch Item ID.....	84
6.5	Time Stamp.....	85
6.6	Authentication .....	85
6.7	Asynchronous Indicator .....	85
6.8	Asynchronous Correlation Value .....	85
6.9	Result Status .....	86
6.10	Result Reason .....	86
6.11	Result Message .....	87
6.12	Batch Order Option.....	87
6.13	Batch Error Continuation Option.....	87
6.14	Batch Count .....	88
6.15	Batch Item.....	88
6.16	Message Extension .....	88
7	Message Format.....	89
7.1	Message Structure.....	89
7.2	Synchronous Operations .....	89
7.3	Asynchronous Operations .....	90
8	Authentication.....	93
9	Message Encoding .....	94
9.1	TTLV Encoding .....	94
9.1.1	TTLV Encoding Fields .....	94
9.1.2	Examples.....	96
9.1.3	Defined Values .....	97
9.2	XML Encoding .....	116
10	Transport .....	117
11	Error Handling .....	118
11.1	General .....	118
11.2	Create .....	119
11.3	Create Key Pair .....	119
11.4	Register.....	120
11.5	Re-key.....	120
11.6	Derive Key .....	121
11.7	Certify.....	121
11.8	Re-certify.....	122
11.9	Locate .....	122
11.10	Check .....	122
11.11	Get.....	123
11.12	Get Attributes.....	123
11.13	Get Attribute List.....	123
11.14	Add Attribute.....	124

11.15	Modify Attribute.....	124
11.16	Delete Attribute.....	125
11.17	Obtain Lease .....	125
11.18	Get Usage Allocation.....	125
11.19	Activate.....	126
11.20	Revoke .....	126
11.21	Destroy .....	126
11.22	Archive.....	127
11.23	Recover .....	127
11.24	Validate.....	127
11.25	Query.....	127
11.26	Cancel .....	127
11.27	Poll.....	127
11.28	Batch Items.....	127
12	Implementation Conformance .....	129
12.1	Conformance clauses for a KMIP Server .....	129
A.	Attribute Cross-reference.....	131
B.	Tag Cross-reference.....	133
C.	Operation and Object Cross-reference .....	138
D.	Acronyms.....	139
E.	List of Figures and Tables .....	142
F.	Acknowledgements.....	149
G.	Revision History .....	151

---

# 1 Introduction

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

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

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

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

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

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

## 1.1 Terminology

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

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

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



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: <ol style="list-style-type: none"> <li>1. The transformation of plaintext data into ciphertext data,</li> <li>2. The transformation of ciphertext data into plaintext data,</li> <li>3. The computation of a digital signature from data,</li> <li>4. The verification of a digital signature,</li> <li>5. The computation of an authentication code from data,</li> <li>6. The verification of an authentication code from data and a received authentication code,</li> </ol>
Decryption	The process of changing ciphertext into plaintext using a cryptographic algorithm and key.
Digest (or hash)	The result of applying a hash function to information.
Digital signature (signature)	The result of a cryptographic transformation of data that, when properly implemented with supporting infrastructure and policy, provides the services of: <ol style="list-style-type: none"> <li>1. origin authentication</li> <li>2. data integrity, and</li> <li>3. signer non-repudiation.</li> </ol>
Encryption	The process of changing plaintext into ciphertext using a cryptographic algorithm and key.
Hash function	A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions satisfy the following properties: <ol style="list-style-type: none"> <li>1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and</li> <li>2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output.</li> </ol>
Integrity	The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner.
Key derivation (derivation)	A function in the lifecycle of keying material; the process by which one or more keys are derived from a shared secret and other information.

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 keys (along with associated integrity information) that provides both confidentiality and integrity protection using a symmetric key.
Message authentication code (MAC)	A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data.
Private key	A cryptographic key, used with a public key cryptographic algorithm, that is uniquely associated with an entity and is not made public. In an asymmetric (public) cryptosystem, the private key is associated with a public key. Depending on the algorithm, the private key may be used to: <ol style="list-style-type: none"> <li>1. Compute the corresponding public key,</li> <li>2. Compute a digital signature that may be verified by the corresponding public key,</li> <li>3. Decrypt data that was encrypted by the corresponding public key, or</li> <li>4. Compute a piece of common shared data, together with other information.</li> </ol>
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 <b>[KMIP-Prof]</b> ).
Public key	A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that may be made public. In an asymmetric (public) cryptosystem, the public key is associated with a private key. The public key may be known by anyone and, depending on the algorithm, may be used to: <ol style="list-style-type: none"> <li>1. Verify a digital signature that is signed by the corresponding private key,</li> <li>2. Encrypt data that can be decrypted by the corresponding private key, or</li> <li>3. Compute a piece of shared data.</li> </ol>
Public key certificate (certificate)	A set of data that uniquely identifies an entity, contains the entity's public key and possibly other information, and is digitally signed by a trusted party, thereby binding the public key to the entity.
Public key cryptographic algorithm	A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible.
Public Key Infrastructure	A framework that is established to issue, maintain and revoke public key certificates.
Recover	To retrieve information that was archived to long-term storage.
Split knowledge	A process by which a cryptographic key is split into $n$ multiple key components, individually providing no knowledge of the original key, which can be subsequently combined to recreate the original cryptographic key. If knowledge of $k$ (where $k$ is less than or equal to $n$ ) components is required to construct the original key, then knowledge of

	any $k-1$ key components provides no information about the original key 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 complement (e.g., encryption and decryption).
X.509 certificate	The ISO/ITU-T X.509 standard defined two types of certificates – the X.509 public key certificate, and the X.509 attribute certificate. Most commonly (including this document), an X.509 certificate refers to the X.509 public key certificate.
X.509 public key certificate	The public key for a user (or device) and a name for the user (or device), together with some other information, rendered un-forgeable by the digital signature of the certification authority that issued the certificate, encoded in the format defined in the ISO/ITU-T X.509 standard.

33

## 34 1.2 Normative References

- 35 **[FIPS186-3]** *Digital Signature Standard (DSS)*, FIPS PUB 186-3, June 2009,  
36 [http://csrc.nist.gov/publications/fips/fips186-3/fips\\_186-3.pdf](http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf)
- 37 **[FIPS197]** *Advanced Encryption Standard*, FIPS PUB 197, Nov 2001,  
38 <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- 39 **[FIPS198-1]** *The Keyed-Hash Message Authentication Code (HMAC)*, FIPS PUB 198-1, July  
40 2008, [http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1\\_final.pdf](http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf)
- 41 **[IEEE1003-1]** IEEE Std 1003.1, *Standard for information technology - portable operating  
42 system interface (POSIX). Shell and utilities*, 2004.
- 43 **[ISO16609]** ISO, *Banking -- Requirements for message authentication using symmetric  
44 techniques*, ISO 16609, 1991
- 45 **[ISO9797-1]** ISO/IEC, *Information technology -- Security techniques -- Message  
46 Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher*,  
47 ISO/IEC 9797-1, 1999.
- 48 **[KMIP-Prof]** OASIS Committee Draft 04, *Key Management Interoperability Protocol Profiles  
49 Version 1.0*, November 2009 [http://docs.oasis-  
50 open.org/kmip/profiles/v1.0/cd04/kmip-profiles-1.0-cd-04.doc](http://docs.oasis-open.org/kmip/profiles/v1.0/cd04/kmip-profiles-1.0-cd-04.doc)
- 51 **[PKCS#1]** RSA Laboratories, *PKCS #1 v2.1: RSA Cryptography Standard*, June 14, 2002.  
52 <http://www.rsa.com/rsalabs/node.asp?id=2125>
- 53 **[PKCS#5]** RSA Laboratories, *PKCS #5 v2.1: Password-Based Cryptography Standard*,  
54 October 5, 2006. <http://www.rsa.com/rsalabs/node.asp?id=2127>
- 55 **[PKCS#7]** RSA Laboratories, *PKCS#7 v1.5: Cryptographic Message Syntax Standard*.  
56 November 1, 1993. <http://www.rsa.com/rsalabs/node.asp?id=2129>
- 57 **[PKCS#8]** RSA Laboratories, *PKCS#8 v1.2: Private-Key Information Syntax Standard*,  
58 November 1, 1993. <http://www.rsa.com/rsalabs/node.asp?id=2130>
- 59 **[PKCS#10]** RSA Laboratories, *PKCS #10 v1.7: Certification Request Syntax Standard*, May  
60 26, 2000. <http://www.rsa.com/rsalabs/node.asp?id=2132>
- 61 **[RFC1319]** B. Kaliski, *The MD2 Message-Digest Algorithm*, IETF RFC 1319, Apr 1992,  
62 <http://www.ietf.org/rfc/rfc1319.txt>
- 63 **[RFC1320]** R. Rivest, *The MD4 Message-Digest Algorithm*, IETF RFC 1320, Apr 1992,  
64 <http://www.ietf.org/rfc/rfc1320.txt>
- 65 **[RFC1321]** R. Rivest, *The MD5 Message-Digest Algorithm*, IETF RFC 1321, Apr 1992,  
66 <http://www.ietf.org/rfc/rfc1321.txt>

- 67 [RFC1421] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I: Message*  
68 *Encryption and Authentication Procedures*, IETF RFC 1421, Feb 1993,  
69 <http://www.ietf.org/rfc/rfc1421.txt>
- 70 [RFC1424] B. Kaliski, *Privacy Enhancement for Internet Electronic Mail: Part IV: Key*  
71 *Certification and Related Services*, IETF RFC 1424, February 1993.  
72 <http://www.ietf.org/rfc/rfc1424.txt>
- 73 [RFC2104] H. Krawczyk, M. Bellare, R. Canetti, *HMAC: Keyed-Hashing for Message*  
74 *Authentication*, IETF RFC 2104. Feb 1007, <http://www.ietf.org/rfc/rfc2104.txt>
- 75 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
76 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 77 [RFC2898] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*,  
78 IETF RFC 2898, Sep 2000, <http://www.ietf.org/rfc/rfc2898.txt>
- 79 [RFC 3394] J. Schaad, R. Housley, *Advanced Encryption Standard (AES) Key Wrap*  
80 *Algorithm*, IETF RFC 3394, Sep 2002, <http://www.ietf.org/rfc/rfc3394.txt>
- 81 [RFC3447] J. Jonsson, B. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA*  
82 *Cryptography Specifications Version 2.1*, IETF RFC 3447 Feb 2003,  
83 <http://www.ietf.org/rfc/rfc3447.txt>
- 84 [RFC3629] F. Yergeau, *UTF-8, a transformation format of ISO 10646*, IETF RFC 3629, Nov  
85 2003, <http://www.ietf.org/rfc/rfc3629.txt>
- 86 [RFC3647] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, *Internet X.509 Public Key*  
87 *Infrastructure Certificate Policy and Certification Practices Framework*, IETF RFC  
88 3647, November 2003. <http://www.ietf.org/rfc/rfc3647.txt>
- 89 [RFC4210] C. Adams, S. Farrell, T. Kause and T. Mononen, *Internet X.509 Public Key*  
90 *Infrastructure Certificate Management Protocol (CMP)*, IETF RFC 2510,  
91 September 2005. <http://www.ietf.org/rfc/rfc4210.txt>
- 92 [RFC4211] J. Schaad, *Internet X.509 Public Key Infrastructure Certificate Request Message*  
93 *Format (CRMF)*, IETF RFC 4211, Sep 2005, <http://www.ietf.org/rfc/rfc4211.txt>
- 94 [RFC4868] S. Kelly, S. Frankel, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-*  
95 *512 with IPsec*, IETF RFC 4868, May 2007, <http://www.ietf.org/rfc/rfc4868.txt>
- 96 [RFC4949] R. Shirey, *Internet Security Glossary, Version 2*, IETF RFC 4949, August 2007.  
97 <http://www.ietf.org/rfc/rfc4949.txt>
- 98 [RFC5272] J. Schaad and M. Meyers, *Certificate Management over CMS (CMC)*, IETF RFC  
99 5272, June 2008. <http://www.ietf.org/rfc/rfc5272.txt>
- 100 [RFC5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, *Internet*  
101 *X.509 Public Key Infrastructure Certificate*, IETF RFC 5280, May 2008,  
102 <http://www.ietf.org/rfc/rfc5280.txt>
- 103 [RFC5649] R. Housley, *Advanced Encryption Standard (AES) Key Wrap with Padding*  
104 *Algorithm*, IETF RFC 5649, Aug 2009, <http://www.ietf.org/rfc/rfc5649.txt>
- 105 [SP800-38A] M. Dworkin, *Recommendation for Block Cipher Modes of Operation – Methods*  
106 *and Techniques*, NIST Special Publication 800-38A, Dec 2001,  
107 <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>
- 108 [SP800-38B] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CMAC*  
109 *Mode for Authentication*, NIST Special Publication 800-38B, May 2005,  
110 [http://csrc.nist.gov/publications/nistpubs/800-38B/SP\\_800-38B.pdf](http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf)
- 111 [SP800-38C] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: the CCM*  
112 *Mode for Authentication and Confidentiality*, NIST Special Publication 800-38C,  
113 May 2004, [http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-](http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf)  
114 [38C\\_updated-July20\\_2007.pdf](http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf)
- 115 [SP800-38D] M. Dworkin, *Recommendation for Block Cipher Modes of Operation:*  
116 *Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication 800-38D, Nov  
117 2007, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>
- 118 [SP800-38E] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The XTS-*  
119 *AES Mode for Confidentiality on Block-Oriented Storage Devices*, NIST Special

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

121

122 **[SP800-57-1]** E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendations for Key Management - Part 1: General (Revised)*, NIST Special Publication 800-57 part 1, March 2007, [http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2\\_Mar08-2007.pdf](http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf)

123

124

125

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

127

128

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

130

131

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

133

134

135

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

137

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

139

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

141

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

143

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

145

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

147

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

149

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

151

152

### 153 **1.3 Non-normative References**

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

155

156

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

158

159

160 **[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>

161

162

163

## 164 2 Objects

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

- 170 • Integer
- 171 • Long Integer
- 172 • Big Integer
- 173 • Enumeration – choices from a predefined list of values
- 174 • Boolean
- 175 • Text String – string of characters representing human-readable text
- 176 • Byte String – sequence of unencoded byte values
- 177 • Date-Time – date and time, with a granularity of one second
- 178 • Interval – time interval expressed in seconds

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

### 180 2.1 Base Objects

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

#### 183 2.1.1 Attribute

184 An Attribute object is a structure (see Table 1) used for sending and receiving Managed Object attributes.  
185 The *Attribute Name* is a text-string that is used to identify the attribute. The *Attribute Index* is an index  
186 number assigned by the key management server when a specified named attribute is allowed to have  
187 multiple instances. The Attribute Index is used to identify the particular instance. Attribute Indices SHALL  
188 start with 0. The Attribute Index of an attribute SHALL NOT change when other instances are added or  
189 deleted. For example, if a particular attribute has 4 instances with Attribute Indices 0, 1, 2 and 3, and the  
190 instance with Attribute Index 2 is deleted, then the Attribute Index of instance 3 is not changed. Attributes  
191 that have a single instance have an Attribute Index of 0, which is assumed if the Attribute Index is not  
192 specified. The *Attribute Value* is either a primitive data type or structured object, depending on the  
193 attribute.

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

194 **Table 1: Attribute Object Structure**

195 **2.1.2 Credential**

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

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

199 **Table 2: Credential Object Structure**

200 **2.1.3 Key Block**

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

- 203 • *Raw* – This is a key that contains only cryptographic key material, encoded as a string of bytes.
- 204 • *Opaque* – This is an encoded key for which the encoding is unknown to the key management  
205 system. It is encoded as a string of bytes.
- 206 • *PKCS1* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#1 object.
- 207 • *PKCS8* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#8 object,  
208 supporting both RSAPrivateKey syntax and EncryptedPrivateKey.
- 209 • *X.509* – This is an encoded object, expressed as a DER-encoded ASN.1 X.509 object.
- 210 • *ECPrivateKey* – This is an ASN.1 encoded elliptic curve private key.
- 211 • *Several Transparent Key types* – These are algorithm-specific structures containing defined  
212 values for the various key types, as defined in Section 2.1.7
- 213 • *Extensions* – These are vendor-specific extensions to allow for proprietary or legacy key formats.

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

216 The Key Block also has the Cryptographic Algorithm and the Cryptographic Length of the key contained  
217 in the Key Value field. Some example values are:

- 218 • RSA keys are typically 1024, 2048 or 3072 bits in length
- 219 • 3DES keys are typically 168 bits in length
- 220 • AES keys are typically 128 or 256 bits in length

221 The Key Block SHALL contain a Key Wrapping Data structure if the key in the Key Value field is wrapped  
222 (i.e., encrypted, or MACed/signed, or both).

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

223

**Table 3: Key Block Object Structure**

## 224 2.1.4 Key Value

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

- 226 • The Key Value structure contains the key material, either as a byte string or as a Transparent Key  
227 structure (see Section 2.1.7 ), and OPTIONAL attribute information that is associated and  
228 encapsulated with the key material. This attribute information differs from the attributes  
229 associated with Managed Objects, and which is obtained via the Get Attributes operation, only by  
230 the fact that it is encapsulated with (and possibly wrapped with) the key material itself.
- 231 • The Key Value Byte String is the wrapped TTLV-encoded (see Section 9.1 ) Key Value structure.



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

232

**Table 4: Key Value Object Structure**

233

### 2.1.5 Key Wrapping Data

234

The Key Block MAY also supply OPTIONAL information about a cryptographic key wrapping mechanism used to wrap the Key Value. This consists of a *Key Wrapping Data* structure (see Table 5). It is only used inside a Key Block.

237

This structure contains fields for:

238

- A *Wrapping Method*, which indicates the method used to wrap the Key Value.

239

- *Encryption Key Information*, which contains the Unique Identifier (see 3.1 ) value of the encryption key and associated cryptographic parameters.

240

241

- *MAC/Signature Key Information*, which contains the Unique Identifier value of the MAC/signature key and associated cryptographic parameters.

242

243

- A *MAC/Signature*, which contains a MAC or signature of the Key Value.

244

- An *IV/Counter/Nonce*, if REQUIRED by the wrapping method.

245

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

252

The following wrapping methods are currently defined:

253

- *Encrypt* only (i.e., encryption using a symmetric key or public key, or authenticated encryption algorithms that use a single key)

254

255

- *MAC/sign* only (i.e., either MACing the Key Value with a symmetric key, or signing the Key Value with a private key)

256

257

- *Encrypt then MAC/sign*

258

- *MAC/sign then encrypt*

259

- *TR-31*

260

- *Extensions*

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

261 **Table 5: Key Wrapping Data Object Structure**

262 The structures of the Encryption Key Information (see Table 6) and the MAC/Signature Key Information  
263 (see Table 7) 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

264 **Table 6: 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 Parameters	Structure, see 3.6	No

265 **Table 7: MAC/Signature Key Information Object Structure**

## 266 2.1.6 Key Wrapping Specification

267 This is a separate structure (see Table 8) that is defined for operations that provide the option to return  
268 wrapped keys. The *Key Wrapping Specification* SHALL be included inside the operation request if clients  
269 request the server to return a wrapped key. If Cryptographic Parameters are specified in the Encryption  
270 Key Information and/or the MAC/Signature Key Information, then the server SHALL verify that they match  
271 one of the instances of the Cryptographic Parameters attribute of the corresponding key. If Cryptographic  
272 Parameters are omitted, then the server SHALL use the Cryptographic Parameters attribute with the  
273 lowest Attribute Index of the corresponding key. If the corresponding key does not have any  
274 Cryptographic Parameters attribute, or if no match is found, then an error is returned.

275 This structure contains:

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

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

282 **Table 8: Key Wrapping Specification Object Structure**

## 283 2.1.7 Transparent Key Structures

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

### 286 2.1.7.1 Transparent Symmetric Key

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

Object	Encoding	REQUIRED
Key Material	Structure	
Key	Byte String	Yes

289 **Table 9: Key Material Object Structure for Transparent Symmetric Keys**

### 290 2.1.7.2 Transparent DSA Private Key

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

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

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

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

296 **2.1.7.3 Transparent DSA Public Key**

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

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

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

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

302 **2.1.7.4 Transparent RSA Private Key**

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

Object	Encoding	REQUIRED
Key Material	Structure	
Modulus	Big Integer	Yes
Private Exponent	Big Integer	No
Public Exponent	Big Integer	No
P	Big Integer	No
Q	Big Integer	No
Prime Exponent P	Big Integer	No
Prime Exponent Q	Big Integer	No
CRT Coefficient	Big Integer	No

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

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

- 307 • Private Exponent
- 308 • P and Q (the first two prime factors of Modulus)
- 309 • Prime Exponent P and Prime Exponent Q.

310 **2.1.7.5 Transparent RSA Public Key**

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

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

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

314 **2.1.7.6 Transparent DH Private Key**

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

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

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

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

320 **2.1.7.7 Transparent DH Public Key**

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

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

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

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

326 **2.1.7.8 Transparent ECDSA Private Key**

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

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

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

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

### 331 2.1.7.9 Transparent ECDSA Public Key

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

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

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

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

### 336 2.1.7.10 Transparent ECDH Private Key

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

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

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

### 340 2.1.7.11 Transparent ECDH Public Key

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

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

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

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

345 **2.1.7.12 Transparent ECMQV Private Key**

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

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

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

349 **2.1.7.13 Transparent ECMQV Public Key**

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

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

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

353 **2.1.8 Template-Attribute Structures**

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

356 The *Template-Attribute*, *Common Template-Attribute*, *Private Key Template-Attribute*, and *Public Key*  
 357 *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

358 **Table 22: Template-Attribute Object Structure**

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

360 **2.2 Managed Objects**

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

## 364 2.2.1 Certificate

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

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

366 **Table 23: Certificate Object Structure**

## 367 2.2.2 Symmetric Key

368 A Managed Cryptographic Object that is a symmetric key.

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

369 **Table 24: Symmetric Key Object Structure**

## 370 2.2.3 Public Key

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

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

373 **Table 25: Public Key Object Structure**

## 374 2.2.4 Private Key

375 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

376 **Table 26: Private Key Object Structure**

## 377 2.2.5 Split Key

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



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

**Table 27: Split Key Object Structure**

384

385 There are three *Split Key Methods* for secret sharing: the first one is based on XOR and the other two are  
 386 based on polynomial secret sharing, according to Adi Shamir, "How to share a secret", Communications  
 387 of the ACM, vol. 22, no. 11, pp. 612-613.

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

389 • When the Split Key Method is XOR, then the Key Material in the Key Value of the Key Block is of  
 390 length  $L$  bits. The number of split keys is Split Key Parts (identical to Split Key Threshold), and  
 391 the secret is reconstructed by XORing all of the parts.

392 • When the Split Key Method is Polynomial Sharing Prime Field, then secret sharing is performed  
 393 in the field  $GF(\text{Prime Field Size})$ , represented as integers, where Prime Field Size is a prime  
 394 bigger than  $2^L$ .

395 • When the Split Key Method is Polynomial Sharing  $GF(2^{16})$ , then secret sharing is performed in  
 396 the field  $GF(2^{16})$ . The Key Material in the Key Value of the Key Block is a bit string of length  $L$ ,  
 397 and when  $L$  is bigger than  $2^{16}$ , then secret sharing is applied piecewise in pieces of 16 bits each.  
 398 The Key Material in the Key Value of the Key Block is the concatenation of the corresponding  
 399 shares of all pieces of the secret.

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

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

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

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

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

408 An element of  $GF(2^8)$  is represented as a byte. Addition and subtraction in  $GF(2^8)$  is performed as  
 409 a bit-wise XOR of the bytes. Multiplication and inversion are more complex (see FIPS PUB 197  
 410 Section 4.1 and 4.2 for details).

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

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

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

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

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

417 The inverse of an element  $uy + v$  is given by  
 418  $(uy + v)^{-1} = ud^{-1}y + (u + v)d^{-1}$ , where  $d = (u + v)v + mu^2$ .

## 419 2.2.6 Template

420 A *Template* is a named Managed Object containing the client-settable attributes of a Managed  
 421 Cryptographic Object (i.e., a stored, named list of attributes). A Template is used to specify the attributes  
 422 of a new Managed Cryptographic Object in various operations. It is intended to be used to specify the  
 423 cryptographic attributes of new objects in a standardized or convenient way. None of the client-settable  
 424 attributes specified in a Template except the Name attribute apply to the template object itself, but instead  
 425 apply to any object created using the Template.

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

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

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

432 Attributes applicable to objects created using the Template are:

- 433 • Cryptographic Algorithm
- 434 • Cryptographic Length
- 435 • Cryptographic Domain Parameters
- 436 • Cryptographic Parameters
- 437 • Operation Policy Name
- 438 • Cryptographic Usage Mask
- 439 • Usage Limits
- 440 • Activation Date
- 441 • Process Start Date
- 442 • Protect Stop Date
- 443 • Deactivation Date
- 444 • Object Group
- 445 • Application Specific Information
- 446 • Contact Information
- 447 • Custom Attribute

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

448 **Table 28: Template Object Structure**

449 **2.2.7 Secret Data**

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

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

453 **Table 29: Secret Data Object Structure**

454 **2.2.8 Opaque Object**

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

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

457 **Table 30: Opaque Object Structure**

458 **3 Attributes**

459 The following subsections describe the attributes that are associated with Managed Objects. These  
 460 attributes are able to be obtained by a client from the server using the Get Attribute operation. Some  
 461 attributes are able to be set by the Add Attribute operation or updated by the Modify Attribute operation,  
 462 and some are able to be deleted by the Delete Attribute operation if they no longer apply to the Managed  
 463 Object.

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

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

470 A server SHALL NOT delete attributes without receiving a request from a client until the object is  
 471 destroyed.

472 The second table (see Table 31) in each subsection lists certain attribute characteristics (e.g., “SHALL  
 473 always have a value”). The “When implicitly set” characteristic indicates which operations (other than  
 474 operations that manage attributes) are able to implicitly add to or modify the attribute of the object, which  
 475 MAY be object(s) on which the operation is performed or object(s) created as a result of the operation.  
 476 Implicit attribute changes MAY occur even if the attribute is not specified in the operation request itself.

SHALL always have a value	All Managed Objects that are of the Object Types for which this attribute applies, SHALL always have this attribute set
Initially set by	Who is permitted to initially set the value of the attribute
Modifiable by server	Is the server allowed to modify the attribute without receiving a request from a client
Modifiable by client	Is the client able to modify 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 cause this attribute to be set without an explicit request from a client
Applies to Object Types	Which Managed Objects MAY have this attribute set

477 **Table 31: Attribute Rules**

478 **3.1 Unique Identifier**

479 The *Unique Identifier* is generated by the key management system to uniquely identify a Managed Object.  
 480 It is only REQUIRED to be unique within the identifier space managed by a single key management  
 481 system, however it is RECOMMENDED that this identifier be globally unique, to allow for key  
 482 management domain export of such objects. This attribute SHALL be assigned by the key management  
 483 system at creation or registration time, and then SHALL NOT be changed or deleted by any entity at any  
 484 time.

Object	Encoding	
Unique Identifier	Text String	

485 **Table 32: 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
Applies to Object Types	All Objects

486 **Table 33: Unique Identifier Attribute Rules**

487 **3.2 Name**

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

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

493 **Table 34: Name Attribute Structure**

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

494

**Table 35: Name Attribute Rules**

### 495 3.3 Object Type

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

Object	Encoding
Object Type	Enumeration, see 9.1.3.2.11

498

**Table 36: Object Type Attribute**

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

499

**Table 37: Object Type Attribute Rules**

### 500 3.4 Cryptographic Algorithm

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

Object	Encoding
Cryptographic Algorithm	Enumeration, see 9.1.3.2.12

503

**Table 38: Cryptographic Algorithm Attribute**

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

504 **Table 39: Cryptographic Algorithm Attribute Rules**

### 505 3.5 Cryptographic Length

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

Object	Encoding	
Cryptographic Length	Integer	

509 **Table 40: Cryptographic Length Attribute**

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

510 **Table 41: Cryptographic Length Attribute Rules**

### 511 3.6 Cryptographic Parameters

512 The *Cryptographic Parameters* attribute is a structure (see Table 42) that contains a set of OPTIONAL  
513 fields that describe certain cryptographic parameters to be used when performing cryptographic  
514 operations using the object. It is possible that specific fields only pertain to certain types of Managed  
515 Cryptographic Objects.

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

516

**Table 42: Cryptographic Parameters Attribute Structure**

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

517

**Table 43: Cryptographic Parameters Attribute Rules**

518

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



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

519

**Table 44: Role Types**

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

### 526 **3.7 Cryptographic Domain Parameters**

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

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

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

532

**Table 45: Cryptographic Domain Parameters Attribute Structure**

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

533

**Table 46: Cryptographic Domain Parameters Attribute Rules**

### 534 3.8 Certificate Type

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

Object	Encoding	
Certificate Type	Enumeration, see 9.1.3.2.6	

537

**Table 47: Certificate Type Attribute**

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

538

**Table 48: Certificate Type Attribute Rules**

### 539 3.9 Certificate Identifier

540 The *Certificate Identifier* attribute is a structure (see Table 49) used to provide the identification of a  
 541 certificate, containing the Issuer Distinguished Name (i.e., from the Issuer field of the certificate) and the  
 542 Certificate Serial Number (i.e., from the Serial Number field of the certificate). This value SHALL be set by  
 543 the server when the certificate is created or registered and then SHALL NOT be changed.

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)

544

**Table 49: Certificate Identifier Attribute Structure**

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

545

**Table 50: Certificate Identifier Attribute Rules**

546

### 3.10 Certificate Subject

547

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

554

If the Subject Alternative Name extension is included in the certificate and is marked *CRITICAL*, 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.

555

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

557

**Table 51: Certificate Subject Attribute Structure**

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

558

**Table 52: Certificate Subject Attribute Rules**

### 559 3.11 Certificate Issuer

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

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

567

**Table 53: Certificate Issuer Attribute Structure**

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

568

**Table 54: Certificate Issuer Attribute Rules**

### 569 3.12 Digest

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

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

576

**Table 55: Digest Attribute Structure**

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	Yes
When implicitly set	Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key
Applies to Object Types	All Cryptographic Objects, Opaque Objects

577

**Table 56: Digest Attribute Rules**

### 578 3.13 Operation Policy Name

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

Object	Encoding	
Operation Policy Name	Text String	

587

**Table 57: Operation Policy Name Attribute**

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

588

**Table 58: Operation Policy Name Attribute Rules**

589 **3.13.1 Operations outside of operation policy control**

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

- 592 • Create
- 593 • Create Key Pair
- 594 • Register
- 595 • Certify
- 596 • Validate
- 597 • Query
- 598 • Cancel
- 599 • Poll

600 **3.13.2 Default Operation Policy**

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

606 **3.13.2.1 Default Operation Policy for Secret Objects**

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

Default Operation Policy for Secret Objects	
Operation	Policy
Re-Key	Allowed to creator only
Derive Key	Allowed to creator only
Locate	Allowed to creator only
Check	Allowed to creator only
Get	Allowed to creator only
Get Attributes	Allowed to creator only
Get Attribute List	Allowed to creator only
Add Attribute	Allowed to creator only
Modify Attribute	Allowed to creator only
Delete Attribute	Allowed to creator only
Obtain Lease	Allowed to creator only

Get Usage Allocation	Allowed to creator only
Activate	Allowed to creator only
Revoke	Allowed to creator only
Destroy	Allowed to creator only
Archive	Allowed to creator only
Recover	Allowed to creator only

**Table 59: Default Operation Policy for Secret Objects**

608  
609 For mandatory profiles, the creator SHALL be the transport-layer identification (see [KMIP-Prof])  
610 provided at the Create or Register operation time.

### 611 **3.13.2.2 Default Operation Policy for Certificates and Public Key Objects**

612 This policy applies to Certificates and Public Keys.

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

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

613  
614 **3.13.2.3 Default Operation Policy for Template Objects**  
615 The operation policy specified as an attribute in the *Create* operation for a template object is the operation  
616 policy used for objects created using that template, and is not the policy used to control operations on the  
617 template itself. There is no mechanism to specify a policy used to control operations on template objects,  
618 so the default policy for template objects is always used for templates created by clients using the  
619 *Register* operation to create template objects.

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

620 **Table 61: Default Operation Policy for Private Template Objects**

621 In addition to private template objects (which are controlled by the above policy, and which MAY be  
622 created by clients or the server), publicly known and usable templates MAY be created and managed by  
623 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

624 **Table 62: Default Operation Policy for Public Template Objects**

### 625 **3.14 Cryptographic Usage Mask**

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

- 629 • Sign
- 630 • Verify
- 631 • Encrypt
- 632 • Decrypt
- 633 • Wrap Key
- 634 • Unwrap Key
- 635 • Export
- 636 • MAC Generate
- 637 • MAC Verify
- 638 • Derive Key
- 639 • Content Commitment
- 640 • Key Agreement
- 641 • Certificate Sign



- 642 • CRL Sign
- 643 • Generate Cryptogram
- 644 • Validate Cryptogram
- 645 • Translate Encrypt
- 646 • Translate Decrypt
- 647 • Translate Wrap
- 648 • Translate Unwrap

649 This list takes into consideration values that MAY appear in the Key Usage extension in an X.509  
 650 certificate. However, the list does not consider the additional usages that MAY appear in the Extended  
 651 Key Usage extension.

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

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

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

655

<b>Object</b>	<b>Encoding</b>	
Cryptographic Usage Mask	Integer	

656 **Table 64: Cryptographic Usage Mask Attribute**

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

657

**Table 65: Cryptographic Usage Mask Attribute Rules**

### 658 3.15 Lease Time

659 The *Lease Time* attribute defines a time interval for a Managed Cryptographic Object beyond which the  
660 client SHALL NOT use the object. This attribute always holds the initial value of a lease, and not the  
661 actual remaining time. Once the lease expires, then the client is only able to renew the lease by calling  
662 Obtain Lease. A server SHALL store in this attribute the maximum Lease Time it is able to serve and a  
663 client obtains the lease time (with Obtain Lease) that is less than or equal to the maximum Lease Time.  
664 This attribute is read-only for clients. It SHALL be modified by the server only.

Object	Encoding	
Lease Time	Interval	

665

**Table 66: Lease Time Attribute**

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

666

**Table 67: Lease Time Attribute Rules**

### 667 3.16 Usage Limits

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

674 fields for two different types of limits, bytes and objects. Exactly one of these two types SHALL be  
 675 present. These fields are:

- 676 • *Usage Limits Total Bytes* – the total number of bytes allowed to be protected. This is the total  
 677 value for the entire life of the object and SHALL NOT be changed once the object begins to be  
 678 used for applying cryptographic protection.
- 679 • *Usage Limits Byte Count* – the currently remaining number of bytes allowed to be protected by  
 680 the object.
- 681 • *Usage Limits Total Objects* – the total number of objects allowed to be protected. This is the total  
 682 value for the entire life of the object and SHALL NOT be changed once the object begins to be  
 683 used for applying cryptographic protection.
- 684 • *Usage Limits Object Count* – the currently remaining number of objects allowed to be protected  
 685 by the object.

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

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

692 **Table 68: Usage Limits Attribute Structure**

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

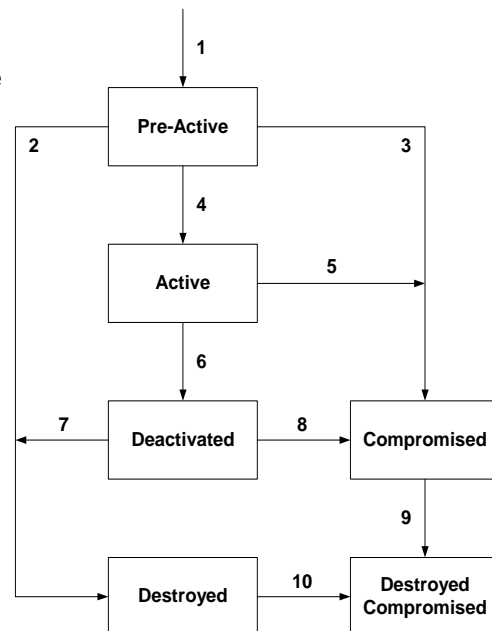
**Table 69: Usage Limits Attribute Rules**

693

### 694 3.17 State

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

- 700 • *Pre-Active*: The object exists but is not yet usable for  
701 any cryptographic purpose.
- 702 • *Active*: The object MAY be used for all cryptographic  
703 purposes that are allowed by its Cryptographic Usage  
704 Mask attribute and, if applicable, by its Process Start  
705 Date (see 3.20 ) and Protect Stop Date (see 3.21 )  
706 attributes.
- 707 • *Deactivated*: The object SHALL NOT be used for  
708 applying cryptographic protection (e.g., encryption or  
709 signing), but, if permitted by the Cryptographic Usage  
710 Mask attribute, then the object MAY be used to  
711 process cryptographically-protected information (e.g.,  
712 decryption or verification), but only under  
713 extraordinary circumstances and when special  
714 permission is granted.
- 715 • *Compromised*: It is possible that the object has been  
716 compromised, and SHOULD only be used to process  
717 cryptographically-protected information in a client that  
718 is trusted to handle compromised cryptographic  
719 objects.
- 720 • *Destroyed*: The object is no longer usable for any  
721 purpose.
- 722 • *Destroyed Compromised*: The object is no longer



**Figure 1: Cryptographic Object States and Transitions**

723 usable for any purpose; however its compromised status MAY be retained for audit or security  
724 purposes.

725 State transitions occur as follows:

- 726 1. The transition from a non-existent key to the Pre-Active state is caused by the creation of the  
727 object. When an object is created or registered, it automatically goes from non-existent to Pre-  
728 Active. If, however, the operation that creates or registers the object contains an Activation Date  
729 that has already occurred, then the state immediately transitions to Active. In this case, the server  
730 SHALL set the Activation Date attribute to the time when the operation is received, or fail the  
731 request attempting to create or register the object, depending on server policy. If the operation  
732 contains an Activation Date attribute in the future, or contains no Activation Date, then the  
733 Cryptographic Object is initialized in the key management system in the Pre-Active state.
- 734 2. The transition from Pre-Active to Destroyed is caused by a client issuing a Destroy operation. The  
735 server destroys the object when (and if) server policy dictates.
- 736 3. The transition from Pre-Active to Compromised is caused by a client issuing a Revoke operation  
737 with a Revocation Reason of Compromised.
- 738 4. The transition from Pre-Active to Active SHALL occur in one of three ways:
  - 739 • The object has an Activation Date in the future. At the time that the Activation Date is  
740 reached, the server changes the state to Active.
  - 741 • A client issues a Modify Attribute operation, modifying the Activation Date to a date in the  
742 past, or the current date. In this case, the server SHALL either set the Activation Date  
743 attribute to the date in the past or the current date, or fail the operation, depending on  
744 server policy.
  - 745 • A client issues an Activate operation on the object. The server SHALL set the Activation  
746 Date to the time the Activate operation is received.
- 747 5. The transition from Active to Compromised is caused by a client issuing a Revoke operation with  
748 a Revocation Reason of Compromised.
- 749 6. The transition from Active to Deactivated SHALL occur in one of three ways:
  - 750 • The object's Deactivation Date is reached.
  - 751 • A client issues a Revoke operation, with a Revocation Reason other than Compromised.
  - 752 • The client issues a Modify Attribute operation, modifying the Deactivation Date to a date in  
753 the past, or the current date. In this case, the server SHALL either set the Deactivation  
754 Date attribute to the date in the past or the current date, or fail the operation, depending on  
755 server policy.
- 756 7. The transition from Deactivated to Destroyed is caused by a client issuing a Destroy operation or  
757 by a server in accordance with server policy. The server destroys the object when (and if) server  
758 policy dictates.
- 759 8. The transition from Deactivated to Compromised is caused by a client issuing a Revoke operation  
760 with a Revocation Reason of Compromised.
- 761 9. The transition from Compromised to Destroyed Compromised is caused by a client issuing a  
762 Destroy operation or by a server in accordance with server policy. The server destroys the object  
763 when (and if) server policy dictates.
- 764 10. The transition from Destroyed to Destroyed Compromised is caused by a client issuing a Revoke  
765 operation with a Revocation Reason of Compromised.

766 Only the transitions described above are permitted.

Object	Encoding	
State	Enumeration, see 9.1.3.2.17	

767

**Table 70: State 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, Certify, Re-certify, Re-key
Applies to Object Types	All Cryptographic Objects

768

**Table 71: State Attribute Rules**

### 769 3.18 Initial Date

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

Object	Encoding	
Initial Date	Date-Time	

774

**Table 72: Initial Date Attribute**

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

775

**Table 73: Initial Date Attribute Rules**

### 776 3.19 Activation Date

777 This is the date and time when the Managed Cryptographic Object MAY begin to be used. This time  
 778 corresponds to state transition 4 (see Section 3.17 ). The object SHALL NOT be used for any  
 779 cryptographic purpose before the *Activation Date* has been reached. Once the state transition has  
 780 occurred, then this attribute SHALL NOT be modified by the server or client.

Object	Encoding	
Activation Date	Date-Time	

781

**Table 74: Activation Date Attribute**

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

782

**Table 75: Activation Date Attribute Rules**

### 783 3.20 Process Start Date

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

Object	Encoding	
Process Start Date	Date-Time	

790

**Table 76: Process Start Date Attribute**

SHALL always have a value	No
Initially set by	Server or Client
Modifiable by server	Yes
Modifiable by client	Yes
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Create, Register, Derive Key, Re-key
Applies to Object Types	Symmetric Keys, Split Keys of symmetric keys, Templates

791

**Table 77: Process Start Date Attribute Rules**

### 792 3.21 Protect Stop Date

793 This is the date and time when a Managed Symmetric Key Object SHALL NOT be used for applying  
 794 cryptographic protection (e.g., encryption or wrapping), depending on the value of its Cryptographic  
 795 Usage Mask attribute. This value MAY be equal to, but SHALL NOT be later than the Deactivation Date.

796 Once the *Protect Stop Date* has occurred, then this attribute SHALL NOT be modified by the server or the  
 797 client.

Object	Encoding	
Protect Stop Date	Date-Time	

798 **Table 78: Protect Stop Date Attribute**

SHALL always have a value	No
Initially set by	Server or Client
Modifiable by server	Yes
Modifiable by client	Yes
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Create, Register, Derive Key, Re-key
Applies to Object Types	Symmetric Keys, Split Keys of symmetric keys, Templates

799 **Table 79: Protect Stop Date Attribute Rules**

## 800 3.22 Deactivation Date

801 The *Deactivation Date* is the date and time when the Managed Cryptographic Object SHALL NOT be  
 802 used for any purpose, except for decryption, signature verification, or unwrapping, but only under  
 803 extraordinary circumstances and only when special permission is granted. This time corresponds to state  
 804 transition 6 (see Section 3.17 ). Once this transition has occurred, then this attribute SHALL NOT be  
 805 modified by the server or client.

Object	Encoding	
Deactivation Date	Date-Time	

806 **Table 80: Deactivation Date Attribute**

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

807 **Table 81: Deactivation Date Attribute Rules**



808 **3.23 Destroy Date**

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

Object	Encoding	
Destroy Date	Date-Time	

813 **Table 82: Destroy Date Attribute**

SHALL always have a value	No
Initially set by	Server
Modifiable by server	No
Modifiable by client	No
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Destroy
Applies to Object Types	All Cryptographic Objects, Opaque Objects

814 **Table 83: Destroy Date Attribute Rules**

815 **3.24 Compromise Occurrence Date**

816 The *Compromise Occurrence Date* is the date and time when the Managed Cryptographic Object was  
 817 first believed to be compromised. If it is not possible to estimate when the compromise occurred, then this  
 818 value SHOULD be set to the Initial Date for the object.

Object	Encoding	
Compromise Occurrence Date	Date-Time	

819 **Table 84: Compromise Occurrence Date Attribute**

SHALL always have a value	No
Initially set by	Server
Modifiable by server	No
Modifiable by client	No
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Revoke
Applies to Object Types	All Cryptographic Objects, Opaque Object

820 **Table 85: Compromise Occurrence Date Attribute Rules**

821 **3.25 Compromise Date**

822 The *Compromise Date* is the date and time when the Managed Cryptographic Object entered into the  
 823 compromised state. This time corresponds to state transitions 3, 5, 8, or 10 (see Section 3.17 ). This time

824 indicates when the key management system was made aware of the compromise, not necessarily when  
 825 the compromise occurred. This attribute is set by the server when it receives a Revoke operation with a  
 826 Revocation Reason of Compromised, or due to server policy or out-of-band administrative action.

Object	Encoding	
Compromise Date	Date-Time	

827 **Table 86: Compromise Date Attribute**

SHALL always have a value	No
Initially set by	Server
Modifiable by server	No
Modifiable by client	No
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Revoke
Applies to Object Types	All Cryptographic Objects, Opaque Object

828 **Table 87: Compromise Date Attribute Rules**

### 829 3.26 Revocation Reason

830 The *Revocation Reason* attribute is a structure (see Table 88) used to indicate why the Managed  
 831 Cryptographic Object was revoked (e.g., “compromised”, “expired”, “no longer used”, etc). This attribute is  
 832 only changed by the server as a part of the Revoke Operation.

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

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

836 **Table 88: Revocation Reason Attribute Structure**

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

837 **Table 89: Revocation Reason Attribute Rules**

838 **3.27 Archive Date**

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

Object	Encoding	
Archive Date	Date-Time	

842 **Table 90: Archive Date 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	Archive
Applies to Object Types	All Objects

843 **Table 91: Archive Date Attribute Rules**

844 **3.28 Object Group**

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

Object	Encoding	
Object Group	Text String	

847 **Table 92: Object Group Attribute**

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

848 **Table 93: Object Group Attribute Rules**

849 **3.29 Link**

850 The *Link* attribute is a structure (see Table 94) used to create a link from one Managed Cryptographic  
 851 Object to another, closely related target Managed Cryptographic Object. The link has a type, and the  
 852 allowed types differ, depending on the Object Type of the Managed Cryptographic Object, as listed  
 853 below. The *Linked Object Identifier* identifies the target Managed Cryptographic Object by its Unique

854 Identifier. The link contains information about the association between the Managed Cryptographic  
 855 Objects (e.g., the private key corresponding to a public key; the parent certificate for a certificate in a  
 856 chain; or for a derived symmetric key, the base key from which it was derived).

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

- 859 • *Private Key Link*. For a Public Key object: the private key corresponding to the public key.
- 860 • *Public Key Link*. For a Private Key object: the public key corresponding to the private key. For a  
 861 Certificate object: the public key contained in the certificate.
- 862 • *Certificate Link*. For Certificate objects: the parent certificate for a certificate in a certificate chain.  
 863 For Public Key objects: the corresponding certificate(s), containing the same public key.
- 864 • *Derivation Base Object Link* for a derived Symmetric Key object: the object(s) from which the  
 865 current symmetric key was derived.
- 866 • *Derived Key Link*: the symmetric key(s) that were derived from the current object.
- 867 • *Replacement Object Link*. For a Symmetric Key object: the key that resulted from the re-key of  
 868 the current key. For a Certificate object: the certificate that resulted from the re-certify. Note that  
 869 there SHALL be only one such replacement object per Managed Object.
- 870 • *Replaced Object Link*. For a Symmetric Key object: the key that was re-keyed to obtain the  
 871 current key. For a Certificate object: the certificate that was re-certified to obtain the current  
 872 certificate.

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

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

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

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

883 **Table 94: Link Attribute Structure**

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

884

**Table 95: Link Attribute Structure Rules**

885 **3.30 Application Specific Information**

886 The *Application Specific Information* attribute is a structure (see Table 96) used to store data specific to  
 887 the application(s) using the Managed Object. It consists of the following fields: an *Application Namespace*  
 888 and *Application Data* specific to that application namespace. A list of standard application namespaces is  
 889 provided in [KMIP-Prof].

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

896

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

897

**Table 96: Application Specific Information Attribute**

898

SHALL always have a value	No
Initially set by	Client or Server (only if the Application Data is omitted, in the client request)
Modifiable by server	Yes (only if the Application Data is omitted in the client request)
Modifiable by client	Yes
Deletable by client	Yes
Multiple instances permitted	Yes
When implicitly set	Re-key, Re-certify
Applies to Object Types	All Objects

899

**Table 97: Application Specific Information Attribute Rules**

900 **3.31 Contact Information**

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

Object	Encoding	
Contact Information	Text String	

903

**Table 98: Contact Information Attribute**

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

904

**Table 99: Contact Information Attribute Rules**

### 905 3.32 Last Change Date

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

Object	Encoding	
Last Change Date	Date-Time	

908

**Table 100: Last Change Date Attribute**

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

909

**Table 101: Last Change Date Attribute Rules**

### 910 3.33 Custom Attribute

911 A *Custom Attribute* is a client- or server-defined attribute intended for vendor-specific purposes. It is  
 912 created by the client and not interpreted by the server, or is created by the server and MAY be interpreted  
 913 by the client. All custom attributes created by the client SHALL adhere to a naming scheme, where the  
 914 name of the attribute SHALL have a prefix of 'x-'. All custom attributes created by the key management  
 915 server SHALL adhere to a naming scheme where the name of the attribute SHALL have a prefix of 'y-'.  
 916 The server SHALL NOT accept a client-created or modified attribute, where the name of the attribute has

917 a prefix of 'y-'. The tag type Custom Attribute is not able to identify the particular attribute; hence such an  
 918 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	The name of the attribute SHALL start with 'x-' or 'y-'.

919 **Table 102 Custom Attribute**

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

920 **Table 103: Custom Attribute Rules**

921

## 4 Client-to-Server Operations

922 The following subsections describe the operations that MAY be requested by a key management client.  
923 Not all clients have to be capable of issuing all operation requests; however any client that issues a  
924 specific request SHALL be capable of understanding the response to the request. All Object Management  
925 operations are issued in requests from clients to servers, and results obtained in responses from servers  
926 to clients. These operations MAY be combined into a batch, which allows multiple operations to be  
927 contained in a single request/response message pair.

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

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

935 The ID Placeholder is obtained from the Unique Identifier returned in response to the Create, Create Pair,  
936 Register, Derive Key, Re-Key, Certify, Re-Certify, Locate, and Recover operations. If any of these  
937 operations successfully completes and returns a Unique Identifier, then the server SHALL copy this  
938 Unique Identifier into the ID Placeholder variable, where it is held until the completion of the operations  
939 remaining in the batched request or until a subsequent operation in the batch causes the ID Placeholder  
940 to be replaced. If the Batch Error Continuation Option is set to Stop and the Batch Order Option is set to  
941 true, then subsequent operations in the batched request MAY make use of the ID Placeholder by omitting  
942 the Unique Identifier field from the request payloads for these operations.

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

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

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

### 4.1 Create

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

959 The request contains information about the type of object being created, and some of the attributes to be  
960 assigned to the object (e.g., Cryptographic Algorithm, Cryptographic Length, etc). This information MAY  
961 be specified by the names of Template objects that already exist.

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



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

964

**Table 104: 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.

965

**Table 105: Create Response Payload**

966  
967

Table 106 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 3.14	Yes

968

**Table 106: Create Attribute Requirements**

969

## 4.2 Create Key Pair

970  
971

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

972  
973  
974  
975  
976  
977

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.

978  
979  
980

A Link Attribute is automatically created by the server for each object, pointing to the corresponding object. The response contains the Unique Identifiers of both created objects. The ID Placeholder value SHALL be set to the Unique Identifier of the Private Key.

Request Payload		
Object	REQUIRED	Description
Common Template-Attribute, see 2.1.8	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.

981 **Table 107: Create Key Pair Request Payload**

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

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

989 If there are multiple templates in the Common, Private, or Public Key Template-Attribute, then the  
990 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 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.

991 **Table 108: Create Key Pair Response Payload**

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

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

994 **Table 109: Create Key Pair Attribute Requirements**

995 **4.3 Register**

996 This operation requests the server to register a Managed Object that was created by the client or  
 997 obtained by the client through some other means, allowing the server to manage the object. The  
 998 arguments in the request are similar to those in the Create operation, but also MAY contain the object  
 999 itself, for storage by the server. Optionally, objects that are not to be stored by the key management  
 1000 system MAY be omitted from the request (e.g., private keys).

1001 The request contains information about the type of object being registered and some of the attributes to  
 1002 be assigned to the object (e.g., Cryptographic Algorithm, Cryptographic Length, etc). This information  
 1003 MAY be specified by the use of a Template-Attribute object.

1004 The response contains the Unique Identifier assigned by the server to the registered object. The server  
 1005 SHALL copy the Unique Identifier returned by this operations into the ID Placeholder variable. The Initial  
 1006 Date attribute of the object SHALL be set to the current time.

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

1007 **Table 110: 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 111: Register Response Payload**

1008  
 1009 If a Managed Cryptographic Object is registered, then the following attributes SHALL be included in the  
 1010 Register request, either explicitly, or via specification of a template that contains the attribute.

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

**Table 112: Register Attribute Requirements**

1011  
 1012 **4.4 Re-key**  
 1013 This request is used to generate a replacement key for an existing symmetric key. It is analogous to the  
 1014 Create operation, except that attributes of the replacement key are copied from the existing key, with the  
 1015 exception of the attributes listed in Table 114.  
 1016 As the replacement key takes over the name attribute of the existing key, Re-key SHOULD only be  
 1017 performed once on a given key.  
 1018 The server SHALL copy the Unique Identifier of the replacement key returned by this operation into the ID  
 1019 Placeholder variable.  
 1020 As a result of Re-key, the Link attribute is set to point to the replacement key.  
 1021 An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date  
 1022 of the replacement key. If Offset is set and dates exist for the existing key, then the dates of the  
 1023 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 ( $AT_2$ ) = $IT_2 + \text{Offset}$
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 = $DT_1 + (AT_2 - AT_1)$

1024

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

1025

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

Attribute	Action
Initial Date, see 3.18	Set to the current time
Destroy Date, see 3.23	Not set
Compromise Occurrence Date, see 3.24	Not set
Compromise Date, see 3.25	Not set
Revocation Reason, see 3.26	Not set
Unique Identifier, see 3.1	New value generated
Usage Limits, see 3.16	The Total Bytes/Total Objects value is copied from the existing key, while the Byte Count/Object Count values are set to the Total Bytes/Total Objects.
Name, see 3.2	Set to the name(s) of the existing key; all name attributes of the existing key are removed.
State, see 3.17	Set based on attributes values, such as dates, as shown in Table 113
Digest, see 3.12	Recomputed from the new key value
Link, see 3.29	Set to point to the existing key as the replaced key
Last Change Date, see 3.32	Set to current time

1026

**Table 114: 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 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 to be created.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

1027

**Table 115: 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.

1028

**Table 116: Re-key Response Payload**

## 1029 4.5 Derive Key

1030 This request is used to derive a symmetric key using a key or secret data that is already known to the key  
 1031 management system. It SHALL only apply to Managed Cryptographic Objects that have the Derive Key  
 1032 bit set in the Cryptographic Usage Mask attribute of the specified Managed Object (i.e., are able to be  
 1033 used for key derivation). If the operation is issued for an object that does not have this bit set, then the  
 1034 server SHALL return an error. For all derivation methods, the client SHALL specify the desired length of  
 1035 the derived key or secret using the Cryptographic Length attribute. If a key is created, then the client  
 1036 SHALL specify both its Cryptographic Length and Cryptographic Algorithm. If the specified length  
 1037 exceeds the output of the derivation method, then the server SHALL return an error. Clients MAY derive  
 1038 multiple keys and IVs by requesting the creation of a Secret Data object and specifying a Cryptographic  
 1039 Length that is the total length of the derived object. The length SHALL NOT exceed the length of the  
 1040 output returned by the chosen derivation method.

1041 The fields in the request specify the Unique Identifiers of the keys or secrets to be used for derivation  
 1042 (e.g., some derivation methods MAY require multiple keys or secrets to derive the result), the method to  
 1043 be used to perform the derivation, and any parameters needed by the specified method. The method is  
 1044 specified as an enumerated value. Currently defined derivation methods include:

- 1045 • *PBKDF2* – This method is used to derive a symmetric key from a password or pass phrase. The  
 1046 PBKDF2 method is published in **[PKCS#5]** and **[RFC2898]**.
- 1047 • *HASH* – This method derives a key by computing a hash over the derivation key or the derivation  
 1048 data.
- 1049 • *HMAC* – This method derives a key by computing an HMAC over the derivation data.
- 1050 • *ENCRYPT* – This method derives a key by encrypting the derivation data.

- 1051 • *NIST800-108-C* – This method derives a key by computing the KDF in Counter Mode as specified  
1052 in **[SP800-108]**.
- 1053 • *NIST800-108-F* – This method derives a key by computing the KDF in Feedback Mode as  
1054 specified in **[SP800-108]**.
- 1055 • *NIST800-108-DPI* – This method derives a key by computing the KDF in Double-Pipeline Iteration  
1056 Mode as specified in **[SP800-108]**.
- 1057 • *Extensions*

1058 The server SHALL perform the derivation function, and then register the derived object as a new  
1059 Managed Object, returning the new Unique Identifier for the new object in the response. The server  
1060 SHALL copy the Unique Identifier returned by this operation into the ID Placeholder variable.

1061 As a result of Derive Key, the Link attributes (i.e., Derived Key Link in the objects from which the key is  
1062 derived, and the Derivation Base Object Link in the derived key) of all objects involved SHALL be set to  
1063 point to the corresponding objects.

Request Payload		
Object	REQUIRED	Description
Object Type, see 3.3	Yes	Determines the type of object to be created.
Unique Identifier, see 3.1	Yes. MAY be repeated	Determines the object or objects to be used to derive a new key. At most, two identifiers MAY be specified: one for the derivation key and another for the secret data. Note that the ID Placeholder SHALL NOT be used here.
Derivation Method, see 9.1.3.2.20	Yes	An Enumeration object specifying the method to be used to derive the new key.
Derivation Parameters, see below	Yes	A Structure object containing the parameters needed by the specified derivation method.
Template-Attribute, see 2.1.8	Yes	Specifies desired object attributes using templates and/or individual attributes; the length and algorithm SHALL always be specified for the creation of a symmetric key.

1064 **Table 117: Derive Key Request Payload**

Response Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly derived 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.

1065 **Table 118: Derive Key Response Payload**

1066 The *Derivation Parameters* for all derivation methods consist of the following parameters, except  
 1067 PBKDF2, which requires two additional parameters.

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

**Table 119: Derivation Parameters Structure (Except PBKDF2)**

1068  
 1069 Cryptographic Parameters identify the Pseudorandom Function (PRF) or the mode of operation of the  
 1070 PRF (e.g., if a key is to be derived using the HASH derivation method, then clients are REQUIRED to  
 1071 indicate the hash algorithm inside Cryptographic Parameters; similarly, if a key is to be derived using AES  
 1072 in CBC mode, then clients are REQUIRED to indicate the Block Cipher Mode). The server SHALL verify  
 1073 that the specified mode matches one of the instances of Cryptographic Parameters set for the  
 1074 corresponding key. If Cryptographic Parameters are omitted, then the server SHALL select the  
 1075 Cryptographic Parameters with the lowest Attribute Index for the specified key. If the corresponding key  
 1076 does not have any Cryptographic Parameters attribute, or if no match is found, then an error is returned.

1077 If a key is derived using HMAC, then the attributes of the derivation key provide enough information about  
 1078 the PRF and the Cryptographic Parameters are ignored.

1079 Derivation Data is either the data to be encrypted, hashed, or HMACed. For the NIST SP 800-108  
 1080 methods [SP800-108], Derivation Data is Label||{0x00}||Context, where the all-zero byte is OPTIONAL.

1081 Most derivation methods (e.g., ENCRYPT) require a derivation key and the derivation data to be  
 1082 encrypted. The HASH derivation method requires either a derivation key or derivation data. Derivation  
 1083 data MAY either be explicitly provided by the client with the Derivation Data field or implicitly provided by  
 1084 providing the Unique Identifier of a Secret Data object. If both are provided, then an error SHALL be  
 1085 returned.

1086 The PBKDF2 derivation method requires two additional parameters:

Object	Encoding	REQUIRED
Derivation Parameters	Structure	Yes
Cryptographic Parameters, see 3.6	Structure	No, depends on the PRF.
Initialization Vector	Byte String	No, depends on the PRF 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 120: PBKDF2 Derivation Parameters Structure**

1087



## 1088 4.6 Certify

1089 This request is used to generate a Certificate object for a public key. This request supports certification of  
 1090 a new public key as well as certification of a public key that has already been certified (i.e., certificate  
 1091 update). Only a single certificate SHALL be requested at a time. Server support for this operation is  
 1092 OPTIONAL, as it requires that the key management system have access to a certification authority (CA).  
 1093 If the server does not support this operation, an error SHALL be returned.

1094 Requests are passed as Byte Strings, which allow multiple certificate request types for X.509 certificates  
 1095 (e.g., PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

1096 The generated Certificate object whose Unique Identifier is returned MAY be obtained by the client via a  
 1097 Get operation in the same batch, using the ID Placeholder mechanism.

1098 As a result of Certify, the Link attribute of the Public Key and of the generated certificate SHALL be set to  
 1099 point at each other.

1100 The server SHALL copy the Unique Identifier of the generated certificate returned by this operation into  
 1101 the ID Placeholder variable.

1102 If the information in the Certificate Request conflicts with the attributes specified in the Template-Attribute,  
 1103 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 is substituted by the server.
Certificate Request Type, see 9.1.3.2.21	Yes	An Enumeration object specifying the type of certificate request.
Certificate Request	Yes	A Byte String object with the certificate request.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

1104 **Table 121: 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.

1105 **Table 122: Certify Response Payload**

## 1106 4.7 Re-certify

1107 This request is used to renew an existing certificate with the same key pair. Only a single certificate  
 1108 SHALL be renewed at a time. Server support for this operation is OPTIONAL, as it requires that the key  
 1109 management system to have access to a certification authority (CA). If the server does not support this  
 1110 operation, an error SHALL be returned.

1111 Requests are passed as Byte Strings, which allow multiple certificate request types for X.509 certificates  
 1112 (e.g., PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

1113 The server SHALL copy the Unique Identifier of the new certificate returned by this operation into the ID  
 1114 Placeholder variable.

1115 If the information in the Certificate Request field in the request conflicts with the attributes specified in the  
 1116 Template-Attribute, then the information in the Certificate Request takes precedence.

1117 As the new certificate takes over the name attribute of the existing certificate, Re-certify SHOULD only be  
 1118 performed once on a given certificate.

1119 The Link attribute of the existing certificate and of the new certificate are set to point at each other. The  
 1120 Link attribute of the Public Key is changed to point to the new certificate.

1121 An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date  
 1122 of the new certificate. If Offset is set, then the dates of the new certificate SHALL be set based on the  
 1123 dates of the existing certificate (if such dates exist) as follows:

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

1124 **Table 123: Computing New Dates from Offset during Re-certify**

1125 Attributes that are not copied from the existing certificate and that are handled in a specific way are:

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

1126 **Table 124: 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 <i>ID Placeholder</i> is substituted by the server.
Certificate Request Type, see 9.1.3.2.21	Yes	An Enumeration object specifying the type of certificate request.
Certificate Request	Yes	A Byte String object with the certificate request.
Offset	No	An Interval object indicating the difference between the Initialization Time of the new certificate and the Activation Date of the new certificate.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

1127

**Table 125: 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.

1128

**Table 126: Re-certify Response Payload**

## 1129 4.8 Locate

1130 This operation requests that the server search for one or more Managed Objects, specified by one or  
 1131 more attributes. All attributes are allowed to be used. However, no attributes specified in the request  
 1132 SHOULD contain Attribute Index values. Attribute Index values SHALL be ignored by the Locate  
 1133 operation. The request MAY also contain a *Maximum Items* field, which specifies the maximum number of  
 1134 objects to be returned. If the Maximum Items field is omitted, then the server MAY return all objects  
 1135 matched, or MAY impose an internal maximum limit due to resource limitations.

1136 If more than one object satisfies the identification criteria specified in the request, then the response MAY  
 1137 contain Unique Identifiers for multiple Managed Objects. Returned objects SHALL match **all** of the  
 1138 attributes in the request. If no objects match, then an empty response payload is returned.

1139 The server returns a list of Unique Identifiers of the found objects, which then MAY be retrieved using the  
 1140 Get operation. If the objects are archived, then the Recover and Get operations are REQUIRED to be  
 1141 used. If a single Unique Identifier is returned to the client, then the server SHALL copy the Unique  
 1142 Identifier returned by this operation into the ID Placeholder variable. If the Locate operation matches  
 1143 more than one object, and the Maximum Items value is omitted in the request, or is set to a value larger  
 1144 than one, then the server SHALL NOT set the ID Placeholder value, causing any subsequent operations  
 1145 that are batched with the Locate, and which do not specify a Unique Identifier explicitly, to fail. This  
 1146 ensures that these batched operations SHALL proceed only if a single object is returned by Locate.

1147 When using the Name or Object Group attributes for identification, wild-cards or regular expressions

1148 (defined, e.g., in [ISO/IEC 9945-2]) MAY be supported by specific key management system  
 1149 implementations.

1150 The Date attributes (e.g., Initial Date, Activation Date, etc) are used to specify a time or a time range. If a  
 1151 single instance of a given Date attribute is used (e.g., the Activation Date), then objects with the same  
 1152 Date attribute are considered to be matching candidate objects. If two instances of the same Date  
 1153 attribute are used (i.e., with two different values specifying a range), then objects for which the Date  
 1154 attribute is inside or at a limit of the range are considered to be matching candidate objects. If a Date  
 1155 attribute is set to its largest possible value, then it is equivalent to an undefined attribute. The KMIP  
 1156 Usage Guide [KMIP-UG] provides examples.

1157 When the Cryptographic Usage Mask attribute is specified in the request, candidate objects are  
 1158 compared against this field via an operation that consists of a logical AND of the requested mask with the  
 1159 mask in the candidate object, and then a comparison of the resulting value with the requested mask. For  
 1160 example, if the request contains a mask value of 10001100010000, and a candidate object mask contains  
 1161 10000100010000, then the logical AND of the two masks is 10000100010000, which is compared against  
 1162 the mask value in the request (10001100010000) and fails the match. This means that a matching  
 1163 candidate object has all of the bits set in its mask that are set in the requested mask, and MAY have  
 1164 additional bits set.

1165 When the Usage Allocation attribute is specified in the request, matching candidate objects SHALL have  
 1166 an Object or Byte Count and Total Objects or Bytes equal to or larger than the values specified in the  
 1167 request.

1168 When an attribute that is defined as a structure is specified, all of the structure fields are not REQUIRED  
 1169 to be specified. For instance, for the Link attribute, if the Linked Object Identifier value is specified without  
 1170 the Link Type value, then matching candidate objects have the Linked Object Identifier as specified,  
 1171 irrespective of their Link Type.

1172 The Storage Status Mask field (see Section 9.1.3.3.2 ) is used to indicate whether only on-line objects,  
 1173 only archived objects, or both on-line and archived objects are to be searched. Note that the server MAY  
 1174 store attributes of archived objects in order to expedite Locate operations that search through archived  
 1175 objects.

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

1176 **Table 127: Locate Request Payload**

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

1177 **Table 128: Locate Response Payload**

## 1178 4.9 Check

1179 This operation requests that the server check for the use of a Managed Object according to values  
1180 specified in the request. This operation SHOULD only be used when placed in a batched set of  
1181 operations, usually following a Locate, Create, Create Pair, Derive Key, Certify, Re-Certify or Re-Key  
1182 operation, and followed by a Get operation. The Unique Identifier field in the request MAY be omitted if  
1183 the operation is in a batched set of operations and follows an operation that sets the ID Placeholder  
1184 variable.

1185 If the server determines that the client is allowed to use the object according to the specified attributes,  
1186 then the server returns the Unique Identifier of the object.

1187 If the server determines that the client is not allowed to use the object according to the specified  
1188 attributes, then the server invalidates the ID Placeholder value and does not return the Unique Identifier,  
1189 and the operation returns the set of attributes specified in the request that caused the server policy denial.  
1190 The only attributes returned are those that resulted in the server determining that the client is not allowed  
1191 to use the object, thus allowing the client to determine how to proceed. The operation also returns a  
1192 failure, and the server SHALL ignore any subsequent operations in the batch.

1193 The additional objects that MAY be specified in the request are limited to:

1194 • Usage Limits Byte Count or Usage Limits Object Count (see Section 3.16 ) – The request MAY  
1195 contain the usage amount that the client deems necessary to complete its needed function. This  
1196 does not require that any subsequent Get Usage Allocation operations request this amount. It  
1197 only means that the client is ensuring that the amount specified is available.

1198 • Cryptographic Usage Mask – This is used to specify the cryptographic operations for which the  
1199 client intends to use the object (see Section 3.14 ). This allows the server to determine if the  
1200 policy allows this client to perform these operations with the object. Note that this MAY be a  
1201 different value from the one specified in a Locate operation that precedes this operation. Locate,  
1202 for example, MAY specify a Cryptographic Usage Mask requesting a key that MAY be used for  
1203 both Encryption and Decryption, but the value in the Check operation MAY specify that the client  
1204 is only using the key for Encryption at this time.

1205 • Lease Time – This specifies a desired lease time (see Section 3.15 ). The client MAY use this to  
1206 determine if the server allows the client to use the object with the specified lease or longer.  
1207 Including this attribute in the Check operation does not actually cause the server to grant a lease,  
1208 but only indicates that the requested lease time value MAY be granted if requested by a  
1209 subsequent, batched, Obtain Lease operation.

1210 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 is substituted by the server.
Usage Limits Byte Count, see 3.16	No	Specifies the number of bytes to be protected to be checked against server policy. SHALL NOT be present if Usage Limits Object Count is present.
Usage Limits Object Count, see 3.16	No	Specifies the number of objects to be protected to be checked against server policy. SHALL NOT be present if Usage Limits Byte Count is present.
Cryptographic Usage Mask, see 3.14	No	Specifies the Cryptographic Usage for which the client intends to use the object.
Lease Time, see 3.15	No	Specifies a Lease Time value that the Client is asking the server to validate against server policy.

1211

**Table 129: Check Request Payload**

Response Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Usage Limits Byte Count, see 3.16	No	Returned by the Server if the Usage Limits value specified in the Request Payload is larger than the value that the server policy allows. SHALL NOT be present if Usage Limits Object Count is present.
Usage Limits Object Count, see 3.16	No	Returned by the Server if the Usage Limits value specified in the Request Payload is larger than the value that the server policy allows. SHALL NOT be present if Usage Limits Byte Count is present.
Cryptographic Usage Mask, see 3.14	No	Returned by the Server if the Cryptographic Usage Mask specified in the Request Payload is rejected by the server for policy violation.
Lease Time, see 3.15	No	Returned by the Server if the Lease Time value in the Request Payload is larger than a valid Lease Time that the server MAY grant.

1212

**Table 130: Check Response Payload**

1213 The encodings of the Usage limits Byte and Object Counts is as shown in Section 3.16

1214 **4.10 Get**

1215 This operation requests that the server returns the Managed Object specified in the request by its Unique  
 1216 Identifier. The Unique Identifier field in the request MAY be omitted if the *Get* operation is in a batched set  
 1217 of operations and follows an operation that sets the ID Placeholder variable.

1218 Only a single object is returned. The response contains the Unique Identifier of the object, along with the  
 1219 object itself, which MAY be wrapped using a wrapping key specified in the request.

1220 The following key format restrictions apply when requesting the server to return an object in a particular  
 1221 format:

- 1222 • If a client registers a key in a given format, the server SHALL be able to return the key during the  
 1223 Get operation in at least that same format as it was registered.
- 1224 • Any other format conversion MAY optionally be supported by the server.

1225

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the object being requested. If omitted, then the ID Placeholder is substituted by the server.
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.

1226

**Table 131: 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

1227

**Table 132: Get Response Payload**

1228 **4.11 Get Attributes**

1229 This operation returns one or more attributes of a Managed Object. The object is specified by its Unique  
 1230 Identifier and the attributes are specified by their name in the request. If a specified attribute has multiple  
 1231 instances, then all instances are returned. If a specified attribute does not exist (i.e., has no value), then it  
 1232 SHALL NOT be present in the returned response. If no requested attributes exist, then the response  
 1233 SHALL consist only of the Unique Identifier.

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 is substituted by the server.
Attribute Name, see 2.1.1	Yes, MAY be repeated	Specifies a desired attribute of the object

1234 **Table 133: 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 for the object

1235 **Table 134: Get Attributes Response Payload**

## 1236 4.12 Get Attribute List

1237 This operation returns a list of the attribute names associated with a Managed Object. The object is  
1238 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 is substituted by the server.

1239 **Table 135: 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 requested attribute names for the object

1240 **Table 136: Get Attribute List Response Payload**

## 1241 4.13 Add Attribute

1242 This request adds a new attribute instance to a Managed Object and sets its value. The request contains  
1243 the Unique Identifier of the Managed Object to which the attribute pertains, and the attribute name and  
1244 value. For non multi-instance attributes, this is how they are created. For multi-instance attributes, this is  
1245 how the first and subsequent values are created. Existing attribute values SHALL only be changed by the  
1246 Modify Attribute operation. Read-Only attributes SHALL NOT be added using the Add Attribute operation.  
1247 No Attribute Index SHALL be specified in the request. The response returns a new Attribute Index if the  
1248 attribute being added is allowed to have multiple instances. Multiple Add Attribute requests MAY be  
1249 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 is substituted by the server.
Attribute, see 2.1.1	Yes	Specifies the attribute of the object to be added.

1250 **Table 137: 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

1251 **Table 138: Add Attribute Response Payload**

## 1252 4.14 Modify Attribute

1253 This request modifies the value of an existing attribute instance associated with a Managed Object. The  
 1254 request contains the Unique Identifier of the Managed Object whose attribute is to be modified, and the  
 1255 attribute name, OPTIONAL Attribute Index, and new value. Only existing attributes MAY be changed via  
 1256 this operation. New attributes SHALL only be added by the Add Attribute operation. Read-Only attributes  
 1257 SHALL NOT be changed using this operation. If an Attribute Index is specified, then only the specified  
 1258 instance is modified. If the attribute has multiple instances, and no Attribute Index is specified in the  
 1259 request, then the Attribute Index is assumed to be 0. If the attribute does not support multiple instances,  
 1260 then the Attribute Index SHALL NOT be specified. Specifying an Attribute Index for which there exists no  
 1261 Attribute Value SHALL result in an error.

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	The Unique Identifier of the object. If omitted, then the ID Placeholder is substituted by the server.
Attribute, see 2.1.1	Yes	Specifies the attribute of the object to be modified.

1262 **Table 139: 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

1263 **Table 140: Modify Attribute Response Payload**

## 1264 4.15 Delete Attribute

1265 This request deletes an attribute associated with a Managed Object. The request contains the Unique  
 1266 Identifier of the Managed Object whose attribute is to be deleted, the attribute name, and optionally the  
 1267 Attribute Index of the attribute. REQUIRED attributes and Read-Only attributes SHALL NOT be deleted  
 1268 by this operation. If no Attribute Index is specified, and the Attribute whose name is specified has multiple

1269 instances, then the operation is rejected. Note that only a single attribute SHALL be deleted at a time.  
 1270 Multiple delete operations (e.g., possibly batched) are necessary to delete several attributes. Attempting  
 1271 to delete a non-existent attribute or specifying an Attribute Index for which there exists no Attribute Value  
 1272 SHALL result in an error.

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 is substituted by the server.
Attribute Name, see 2.1.1	Yes	Specifies the name of the attribute to be deleted.
Attribute Index, see 2.1.1	No	Specifies the Index of the Attribute.

1273 **Table 141: 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

1274 **Table 142: Delete Attribute Response Payload**

## 1275 4.16 Obtain Lease

1276 This request is used to obtain a new *Lease Time* for a specified Managed Object. The Lease Time is an  
 1277 interval value that determines when the client's internal cache of information about the object expires and  
 1278 needs to be renewed. If the returned value of the lease time is zero, then the server is indicating that no  
 1279 lease interval is effective, and the client MAY use the object without any lease time limit. If a client's lease  
 1280 expires, then the client SHALL NOT use the associated cryptographic object until a new lease is  
 1281 obtained. If the server determines that a new lease SHALL NOT be issued for the specified cryptographic  
 1282 object, then the server SHALL respond to the Obtain Lease request with an error.

1283 The response payload for the operation also contains the current value of the Last Change Date attribute  
 1284 for the object. This MAY be used by the client to determine if any of the attributes cached by the client  
 1285 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 <i>ID Placeholder</i> is substituted by the server.

1286 **Table 143: Obtain Lease Request Payload**

Response Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Lease Time, see 3.15	Yes	An interval (in seconds) that specifies the amount of time that the object MAY be used until a new lease needs to be obtained.
Last Change Date, see 3.32	Yes	The date and time indicating when the latest change was made to the contents or any attribute of the specified object.

1287

**Table 144: Obtain Lease Response Payload**

## 1288 4.17 Get Usage Allocation

1289 This request is used to obtain an allocation from the current Usage Limits values to allow the client to use  
 1290 the Managed Cryptographic Object for applying cryptographic protection. The allocation only applies to  
 1291 Managed Cryptographic Objects that are able to be used for applying protection (e.g., symmetric keys for  
 1292 encryption, private keys for signing, etc.) and is only valid if the Managed Cryptographic Object has a  
 1293 Usage Limits attribute. Usage for processing cryptographically-protected information (e.g., decryption,  
 1294 verification, etc.) is not limited and is not able to be allocated. A Managed Cryptographic Object that has a  
 1295 Usage Limits attribute SHALL NOT be used by a client for applying cryptographic protection unless an  
 1296 allocation has been obtained using this operation. The operation SHALL only be requested during the  
 1297 time that protection is enabled for these objects (i.e., after the Activation Date and before the Protect Stop  
 1298 Date). If the operation is requested for an object that has no Usage Limits attribute, or is not an object that  
 1299 MAY be used for applying cryptographic protection, then the server SHALL return an error.

1300 The fields in the request specify the number of bytes or number of objects that the client needs to protect.  
 1301 Exactly one of the two count fields SHALL be specified in the request. If the requested amount is not  
 1302 available or if the Managed Object is not able to be used for applying cryptographic protection at this time,  
 1303 then the server SHALL return an error. The server SHALL assume that the entire allocated amount has  
 1304 been consumed. Once the entire allocated amount has been consumed, the client SHALL NOT continue  
 1305 to use the Managed Cryptographic Object for applying cryptographic protection until a new allocation is  
 1306 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 Byte Count, see 3.16	No	The number of bytes to be protected. SHALL be present if Usage Limits Object Count is not present.
Usage Limits Object Count, see 3.16	No	The number of objects to be protected. SHALL be present if Usage Limits Byte Count is not present.

1307

**Table 145: Get Usage Allocation Request Payload**

Response Payload		
Object	REQUIRED	Description

Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
----------------------------	-----	--------------------------------------

1308

**Table 146: Get Usage Allocation Response Payload**

## 1309 4.18 Activate

1310 This request is used to activate a Managed Cryptographic Object. The request SHALL NOT specify a  
 1311 Template object. The request contains the Unique Identifier of the Managed Cryptographic Object. The  
 1312 operation SHALL only be performed on an object in the Pre-Active state and has the effect of changing its  
 1313 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 is substituted by the server.

1314

**Table 147: Activate Request Payload**

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

1315

**Table 148: Activate Response Payload**

## 1316 4.19 Revoke

1317 This request is used to revoke a Managed Cryptographic Object or an Opaque Object. The request  
 1318 SHALL NOT specify a Template object. The request contains the unique identifier of the Managed  
 1319 Cryptographic Object and a reason for the revocation (e.g., “compromised”, “no longer used”, etc).  
 1320 Special authentication and authorization SHOULD be enforced to perform this request (see [KMIP-UG]).  
 1321 Only the object creator or an authorized security officer SHOULD be allowed to issue this request. The  
 1322 operation has one of two effects. If the revocation reason is “compromised”, then the object is placed into  
 1323 the “compromised” state, and the Compromise Date attribute is set to the current date and time.  
 1324 Otherwise, the object is placed into the “deactivated” state, and the Deactivation Date attribute is set to  
 1325 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 is substituted by the server.
Revocation Reason, see 3.26	Yes	Specifies the reason for revocation.
Compromise Occurrence Date, see 3.24	No	SHALL be specified if the Revocation Reason is 'compromised'.

1326

**Table 149: Revoke Request Payload**

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

1327

**Table 150: Revoke Response Payload**

## 1328 4.20 Destroy

1329 This request is used to indicate to the server that the key material for the specified Managed Object  
1330 SHALL be destroyed. The meta-data for the key material MAY be retained by the server (e.g., used to  
1331 ensure that an expired or revoked private signing key is no longer available). Special authentication and  
1332 authorization SHOULD be enforced to perform this request (see [KMIP-UG]). Only the object creator or  
1333 an authorized security officer SHOULD be allowed to issue this request. If the Unique Identifier specifies  
1334 a Template object, then the object itself, including all meta-data, SHALL be destroyed.

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

1335 **Table 151: Destroy Request Payload**

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

1336 **Table 152: Destroy Response Payload**

## 1337 4.21 Archive

1338 This request is used to specify that a Managed Object MAY be archived. The actual time when the object  
1339 is archived, the location of the archive, or level of archive hierarchy is determined by the policies within  
1340 the key management system and is not specified by the client. The request contains the unique identifier  
1341 of the Managed Object. Special authentication and authorization SHOULD be enforced to perform this  
1342 request (see [KMIP-UG]). Only the object creator or an authorized security officer SHOULD be allowed to  
1343 issue this request. This request is only a "hint" to the key management system to possibly archive the  
1344 object.

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

1345 **Table 153: Archive Request Payload**

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

1346 **Table 154: Archive Response Payload**

## 1347 4.22 Recover

1348 This request is used to obtain access to a Managed Object that has been archived. This request MAY  
1349 require asynchronous polling to obtain the response due to delays caused by retrieving the object from  
1350 the archive. Once the response is received, the object is now on-line, and MAY be obtained (e.g., via a  
1351 Get operation). Special authentication and authorization SHOULD be enforced to perform this request  
1352 (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 is substituted by the server.

1353 **Table 155: Recover Request Payload**

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

1354 **Table 156: Recover Response Payload**

### 1355 4.23 Validate

1356 This requests that the server validate a certificate chain and return information on its validity. Only a  
 1357 single certificate chain SHALL be included in each request. Support for this operation at the server is  
 1358 OPTIONAL. If the server does not support this operation, an error SHALL be returned.

1359 The request may contain a list of certificate objects, and/or a list of Unique Identifiers that identify  
 1360 Managed Certificate objects. Together, the two lists compose a certificate chain to be validated. The  
 1361 request MAY also contain a date for which the certificate chain is REQUIRED to be valid.

1362 The method or policy by which validation is conducted is a decision of the server and is outside of the  
 1363 scope of this protocol. Likewise, the order in which the supplied certificate chain is validated and the  
 1364 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 is valid.

1365 **Table 157: Validate Request Payload**

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

1366 **Table 158: Validate Response Payload**

### 1367 4.24 Query

1368 This request is used by the client to interrogate the server to determine its capabilities and/or protocol  
 1369 mechanisms. The *Query* operation SHOULD be invocable by unauthenticated clients to interrogate server  
 1370 features and functions. The *Query Function* field in the request SHALL contain one or more of the  
 1371 following items:

- 1372 • Query Operations
- 1373 • Query Objects
- 1374 • Query Server Information
- 1375 • Query Application Namespaces

1376 The *Operation* fields in the response contain Operation enumerated values, which SHALL list the  
 1377 OPTIONAL operations that the server supports. If the request contains a Query Operations value in the  
 1378 Query Function field, then these fields SHALL be returned in the response. The OPTIONAL operations  
 1379 are:

- 1380 • Validate
- 1381 • Certify
- 1382 • Re-Certify
- 1383 • Notify
- 1384 • Put

1385 The *Object Type* fields in the response contain Object Type enumerated values, which SHALL list the  
 1386 object types that the server supports. If the request contains a *Query Objects* value in the Query Function  
 1387 field, then these fields SHALL be returned in the response. The object types (any of which are  
 1388 OPTIONAL) are:

- 1389 • Certificate
- 1390 • Symmetric Key
- 1391 • Public Key
- 1392 • Private Key
- 1393 • Split Key
- 1394 • Template
- 1395 • Secret Data
- 1396 • Opaque Object

1397 The *Server Information* field in the response is a structure containing vendor-specific fields and/or  
 1398 substructures. If the request contains a *Query Server Information* value in the Query Function field, then  
 1399 this field SHALL be returned in the response.

1400 The Application Namespace fields in the response contain the namespaces that the server SHALL  
 1401 generate values for if requested by the client (see Section 3.30 ). These fields SHALL only be returned in  
 1402 the response if the request contains a Query Application Namespaces value in the Query Function field.

1403 Note that the response payload is empty if there are no values to return.

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

1404 **Table 159: Query Request Payload**

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

1405 **Table 160: Query Response Payload**

## 1406 4.25 Cancel

1407 This request is used to cancel an outstanding asynchronous operation. The correlation value (see Section  
 1408 6.8 ) of the original operation SHALL be specified in the request. The server SHALL respond with a  
 1409 *Cancellation Result* that contains one of the following values:

- 1410 • *Canceled* – The cancel operation succeeded in canceling the pending operation.
- 1411 • *Unable To Cancel* – The cancel operation is unable to cancel the pending operation.
- 1412 • *Completed* – The pending operation completed successfully before the cancellation operation  
 1413 was able to cancel it.
- 1414 • *Failed* – The pending operation completed with a failure before the cancellation operation was  
 1415 able to cancel it.
- 1416 • *Unavailable* – The specified correlation value did not match any recently pending or completed  
 1417 asynchronous operations.

1418 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

1419 **Table 161: Cancel Request Payload**

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

1420 **Table 162: Cancel Response Payload**



1421 **4.26 Poll**

1422 This request is used to poll the server in order to obtain the status of an outstanding asynchronous  
1423 operation. The correlation value (see Section 6.8 ) of the original operation SHALL be specified in the  
1424 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

1425 **Table 163: Poll Request Payload**

1426 The server SHALL reply with one of two responses:

1427 If the operation has not completed, the response SHALL contain no payload and a Result Status of  
1428 Pending.

1429 If the operation has completed, the response SHALL contain the appropriate payload for the operation.  
1430 This response SHALL be identical to the response that would have been sent if the operation had  
1431 completed synchronously.

## 1432 5 Server-to-Client Operations

1433 Server-to-client operations are used by servers to send information or Managed Cryptographic Objects to  
1434 clients via means outside of the normal client-server request-response mechanism. These operations are  
1435 used to send Managed Cryptographic Objects directly to clients without a specific request from the client.

### 1436 5.1 Notify

1437 This operation is used to notify a client of events that resulted in changes to attributes of an object. This  
1438 operation is only ever sent by a server to a client via means outside of the normal client request/response  
1439 protocol, using information known to the server via unspecified configuration or administrative  
1440 mechanisms. It contains the Unique Identifier of the object to which the notification applies, and a list of  
1441 the attributes whose changed values have triggered the notification. The message is sent as a normal  
1442 Request message, except that the Maximum Response Size, Asynchronous Indicator, Batch Error  
1443 Continuation Option, and Batch Order Option fields are not allowed. The client SHALL send a response in  
1444 the form of a Response Message containing no payload, unless both the client and server have prior  
1445 knowledge (obtained via out-of-band mechanisms) that the client is not able to respond. Server and Client  
1446 support for this message is OPTIONAL.

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.

1447 **Table 164: Notify Message Payload**

### 1448 5.2 Put

1449 This operation is used to “push” Managed Cryptographic Objects to clients. This operation is only ever  
1450 sent by a server to a client via means outside of the normal client request/response protocol, using  
1451 information known to the server via unspecified configuration or administrative mechanisms. It contains  
1452 the Unique Identifier of the object that is being sent, and the object itself. The message is sent as a  
1453 normal Request message, except that the Maximum Response Size, Asynchronous Indicator, Batch Error  
1454 Continuation Option, and Batch Order Option fields are not allowed. The client SHALL send a response in  
1455 the form of a Response Message containing no payload, unless both the client and server have prior  
1456 knowledge (obtained via out-of-band mechanisms) that the client is not able to respond. Server and client  
1457 support for this message is OPTIONAL.

1458 The *Put Function* field indicates whether the object being “pushed” is a new object, or is a replacement for  
1459 an object already known to the client (e.g., when pushing a certificate to replace one that is about to  
1460 expire, the Put Function field would be set to indicate replacement, and the Unique Identifier of the  
1461 expiring certificate would be placed in the *Replaced Unique Identifier* field). The Put Function SHALL  
1462 contain one of the following values:

- 1463 • *New* – which indicates that the object is not a replacement for another object.
- 1464 • *Replace* – which indicates that the object is a replacement for another object, and that the  
1465 *Replaced Unique Identifier* field is present and contains the identification of the replaced object.

1466 The Attribute field contains one or more attributes that the server is sending along with the object. The  
1467 server MAY include attributes with the object to specify how the object is to be used by the client. The  
1468 server MAY include a Lease Time attribute that grants a lease to the client.

1469 If the Managed Object is a wrapped key, then the key wrapping specification SHALL be exchanged prior  
 1470 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 9.1.3.2.25	Yes	Indicates function for Put message.
Replaced Unique Identifier, see 3.1	No	Unique Identifier of the replaced object. SHALL be present if the <i>Put Function</i> is <i>Replace</i> .
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.

1471

**Table 165: Put Message Payload**

## 1472 6 Message Contents

1473 The messages in the protocol consist of a message header, one or more batch items (which contain  
1474 OPTIONAL message payloads), and OPTIONAL message extensions. The message headers contain  
1475 fields whose presence is determined by the protocol features used (e.g., asynchronous responses). The  
1476 field contents are also determined by whether the message is a request or a response. The message  
1477 payload is determined by the specific operation being requested or to which is being replied.

1478 The message headers are structures that contain some of the following objects.

### 1479 6.1 Protocol Version

1480 This field contains the version number of the protocol, ensuring that the protocol is fully understood by  
1481 both communicating parties. The version number is specified in two parts, major and minor. Servers and  
1482 clients SHALL support backward compatibility with versions of the protocol with the same major version.  
1483 Support for backward compatibility with different major versions is OPTIONAL.

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

1484 **Table 166: Protocol Version Structure in Message Header**

### 1485 6.2 Operation

1486 This field indicates the operation being requested or the operation for which the response is being  
1487 returned. The operations are defined in Sections 4 and 5

Object	Encoding	
Operation	Enumeration, see 9.1.3.2.26	

1488 **Table 167: Operation in Batch Item**

### 1489 6.3 Maximum Response Size

1490 This field is optionally contained in a request message, and is used to indicate the maximum size of a  
1491 response that the requester SHALL handle. It SHOULD only be sent in requests that possibly return large  
1492 replies.

Object	Encoding	
Maximum Response Size	Integer	

1493 **Table 168: Maximum Response Size in Message Request Header**

### 1494 6.4 Unique Batch Item ID

1495 This field is optionally contained in a request, and is used for correlation between requests and  
1496 responses. If a request has a *Unique Batch Item ID*, then responses to that request SHALL have the  
1497 same Unique Batch Item ID.

Object	Encoding	
Unique Batch Item ID	Byte String	

1498

**Table 169: Unique Batch Item ID in Batch Item**

## 1499 6.5 Time Stamp

1500 This field is optionally contained in a request, is REQUIRED in a response, is used for time stamping, and  
 1501 MAY be used to enforce reasonable time usage at a client (e.g., a server MAY choose to reject a request  
 1502 if a client's time stamp contains a value that is too far off the known correct time). Note that the time  
 1503 stamp MAY be used by a client that has no real-time clock, but has a countdown timer, to obtain useful  
 1504 "seconds from now" values from all of the Date attributes by performing a subtraction.

Object	Encoding	
Time Stamp	Date-Time	

1505

**Table 170: Time Stamp in Message Header**

## 1506 6.6 Authentication

1507 This is used to authenticate the requester. It is an OPTIONAL information item, depending on the type of  
 1508 request being issued and on server policies. Servers MAY require authentication on no requests, a  
 1509 subset of the requests, or all requests, depending on policy. Query operations used to interrogate server  
 1510 features and functions SHOULD NOT require authentication.

1511 The authentication mechanisms are described and discussed in Section 8 .

Object	Encoding	REQUIRED
Authentication	Structure	
Credential	Structure, see 2.1.2	Yes

1512

**Table 171: Authentication Structure in Message Header**

## 1513 6.7 Asynchronous Indicator

1514 This Boolean flag indicates whether the client is able to accept an asynchronous response. It SHALL  
 1515 have the Boolean value True if the client is able to handle asynchronous responses, and the value False  
 1516 otherwise. If not present in a request, then False is assumed. If a client indicates that it is not able to  
 1517 handle asynchronous responses (i.e., flag is set to False), and the server is not able to process the  
 1518 request synchronously, then the server SHALL respond to the request with a failure.

Object	Encoding	
Asynchronous Indicator	Boolean	

1519

**Table 172: Asynchronous Indicator in Message Request Header**

## 1520 6.8 Asynchronous Correlation Value

1521 This is returned in the immediate response to an operation that requires asynchronous polling. Note: the  
 1522 server decides which operations are performed synchronously or asynchronously. A server-generated  
 1523 correlation value SHALL be specified in any subsequent Poll or Cancel operations that pertain to the  
 1524 original operation.

Object	Encoding	
Asynchronous Correlation Value	Byte String	

1525

**Table 173: Asynchronous Correlation Value in Response Batch Item**

## 1526 6.9 Result Status

1527 This is sent in a response message and indicates the success or failure of a request. The following values  
1528 MAY be set in this field:

- 1529 • *Success* – The requested operation completed successfully.
- 1530 • *Pending* – The requested operation is in progress, and it is necessary to obtain the actual result  
1531 via asynchronous polling. The asynchronous correlation value SHALL be used for the subsequent  
1532 polling of the result status.
- 1533 • *Undone* – The requested operation was performed, but had to be undone (i.e., due to a failure in  
1534 a batch for which the Error Continuation Option was set to Undo).
- 1535 • *Failure* – The requested operation failed.

Object	Encoding	
Result Status	Enumeration, see 9.1.3.2.27	

1536 **Table 174: Result Status in Response Batch Item**

## 1537 6.10 Result Reason

1538 This field indicates a reason for failure or a modifier for a partially successful operation and SHALL be  
1539 present in responses that return a Result Status of Failure. In such a case the Result Reason SHALL be  
1540 set as specified in Section 11 . It is OPTIONAL in any response that returns a Result Status of Success.  
1541 The following defined values are defined for this field:

- 1542 • *Item not found* – A requested object was not found or did not exist.
- 1543 • *Response too large* – The response to a request would exceed the *Maximum Response Size* in  
1544 the request.
- 1545 • *Authentication not successful* – The authentication information in the request was not able to be  
1546 validated, or there was no authentication information in the request when there SHOULD have  
1547 been.
- 1548 • *Invalid message* – The request message was not understood by the server.
- 1549 • *Operation not supported* – The operation requested by the request message is not supported by  
1550 the server.
- 1551 • *Missing data* – The operation requires additional OPTIONAL information in the request, which  
1552 was not present.
- 1553 • *Invalid field* – Some data item in the request has an invalid value.
- 1554 • *Feature not supported* – An OPTIONAL feature specified in the request is not supported.
- 1555 • *Operation canceled by requester* – The operation was asynchronous, and the operation was  
1556 canceled by the Cancel operation before it completed successfully.
- 1557 • *Cryptographic failure* – The operation failed due to a cryptographic error.
- 1558 • *Illegal operation* – The client requested an operation that was not able to be performed with the  
1559 specified parameters.
- 1560 • *Permission denied* – The client does not have permission to perform the requested operation.
- 1561 • *Object archived* – The object SHALL be recovered from the archive before performing the  
1562 operation.
- 1563 • *General failure* – The request failed for a reason other than the defined reasons above.

Object	Encoding	
Result Reason	Enumeration, see 9.1.3.2.28	

1564 **Table 175: Result Reason in Response Batch Item**

## 1565 6.11 Result Message

1566 This field MAY be returned in a response. It contains a more descriptive error message, which MAY be  
 1567 used by the client to display to an end user or for logging/auditing purposes.

Object	Encoding	
Result Message	Text String	

1568 **Table 176: Result Message in Response Batch Item**

## 1569 6.12 Batch Order Option

1570 A Boolean value used in requests where the Batch Count is greater than 1. If True, then batched  
 1571 operations SHALL be executed in the order in which they appear within the request. If False, then the  
 1572 server MAY choose to execute the batched operations in any order. If not specified, then False is  
 1573 assumed (i.e., no implied ordering). Server support for this feature is OPTIONAL, but if the server does  
 1574 not support the feature, and a request is received with the batch order option set to True, then the entire  
 1575 request SHALL be rejected.

Object	Encoding	
Batch Order Option	Boolean	

1576 **Table 177: Batch Order Option in Message Request Header**

## 1577 6.13 Batch Error Continuation Option

1578 This option SHALL only be present if the Batch Count is greater than 1. This option SHALL have one of  
 1579 three values:

- 1580 • *Undo* – If any operation in the request fails, then the server SHALL undo all the previous  
 1581 operations.
- 1582 • *Stop* – If an operation fails, then the server SHALL NOT continue processing subsequent  
 1583 operations in the request. Completed operations SHALL NOT be undone.
- 1584 • *Continue* – Return an error for the failed operation, and continue processing subsequent  
 1585 operations in the request.

1586 If not specified, then Stop is assumed.

1587 Server support for this feature is OPTIONAL, but if the server does not support the feature, and a request  
 1588 is received containing the *Batch Error Continuation* option with a value other than the default Stop, then  
 1589 the entire request SHALL be rejected.

Object	Encoding	
Batch Error Continuation Option	Enumeration, see 9.1.3.2.29	

1590 **Table 178: Batch Error Continuation Option in Message Request Header**

1591 **6.14 Batch Count**

1592 This field contains the number of Batch Items in a message and is REQUIRED. If only a single operation  
 1593 is being requested, then the batch count SHALL be set to 1. The Message Payload, which follows the  
 1594 Message Header, contains one or more batch items.

Object	Encoding	
Batch Count	Integer	

1595 **Table 179: Batch Count in Message Header**

1596 **6.15 Batch Item**

1597 This field consists of a structure that holds the individual requests or responses in a batch, and is  
 1598 REQUIRED. The contents of the batch items are described in Sections 7.2 and 7.3 .

Object	Encoding	
Batch Item	Structure	

1599 **Table 180: Batch Item in Message**

1600 **6.16 Message Extension**

1601 The *Message Extension* is an OPTIONAL structure that MAY be appended to any Batch Item. It is used  
 1602 to extend protocol messages for the purpose of adding vendor specified extensions. The Message  
 1603 Extension is a structure containing a Vendor Identification, a Criticality Indicator, and vendor-specific  
 1604 extensions. The *Vendor Identification* SHALL be a text string that uniquely identifies the vendor, allowing  
 1605 a client to determine if it is able to parse and understand the extension. If a client or server receives a  
 1606 protocol message containing a message extension that it does not understand, then its actions depend  
 1607 on the *Criticality Indicator*. If the indicator is True (i.e., Critical), and the receiver does not understand the  
 1608 extension, then the receiver SHALL reject the entire message. If the indicator is False (i.e., Non-Critical),  
 1609 and the receiver does not understand the extension, then the receiver MAY process the rest of the  
 1610 message as if the extension were not present.

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

1611 **Table 181: Message Extension Structure in Batch Item**



1612 **7 Message Format**

1613 Messages contain the following objects and fields. All fields SHALL appear in the order specified.

1614 **7.1 Message Structure**

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

1615 **Table 182: Request Message Structure**

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

1616 **Table 183: Response Message Structure**

1617 **7.2 Synchronous Operations**

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

1618 **Table 184: Synchronous Request Header Structure**

Synchronous 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 <i>Batch Count</i> > 1, see 6.4
Request Payload	Yes	Structure, contents depend on the Operation, see 4 and 5
Message Extension	No	See 6.16

1619

**Table 185: Synchronous Request Batch Item Structure**

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

1620

**Table 186: Synchronous Response Header Structure**

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

1621

**Table 187: Synchronous Response Batch Item Structure**

### 1622 7.3 Asynchronous Operations

1623 If the client is capable of accepting asynchronous responses, then it MAY set the *Asynchronous Indicator*  
 1624 in the header of a batched request. The batched responses MAY contain a mixture of synchronous and  
 1625 asynchronous responses.

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

1626

**Table 188: Asynchronous Request Header Structure**

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

1627

**Table 189: Asynchronous Request Batch Item Structure**

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

1628

**Table 190: Asynchronous Response Header Structure**

Asynchronous Response Batch Item		
----------------------------------	--	--

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

**Table 191: Asynchronous Response Batch Item Structure**

1629

---

1630

## 8 Authentication

1631 The mechanisms used to authenticate the client to the server and the server to the client are not part of  
1632 the message definitions, and are external to the protocol. The KMIP Server SHALL support authentication  
1633 as defined in **[KMIP-Prof]**.

---

## 1634 9 Message Encoding

1635 To support different transport protocols and different client capabilities, a number of message-encoding  
1636 mechanisms are supported.

### 1637 9.1 TTLV Encoding

1638 In order to minimize the resource impact on potentially low-function clients, one encoding mechanism to  
1639 be used for protocol messages is a simplified TTLV (Tag, Type, Length, Value) scheme.

1640 The scheme is designed to minimize the CPU cycle and memory requirements of clients that need to  
1641 encode or decode protocol messages, and to provide optimal alignment for both 32-bit and 64-bit  
1642 processors. Minimizing bandwidth over the transport mechanism is considered to be of lesser importance.

#### 1643 9.1.1 TTLV Encoding Fields

1644 Every Data object encoded by the TTLV scheme consists of four items, in order:

##### 1645 9.1.1.1 Item Tag

1646 An Item Tag is a three-byte binary unsigned integer, transmitted big endian, which contains a number that  
1647 designates the specific Protocol Field or Object that the TTLV object represents. To ease debugging, and  
1648 to ensure that malformed messages are detected more easily, all tags SHALL contain either the value 42  
1649 in hex or the value 54 in hex as the high order (first) byte. Tags defined by this specification contain hex  
1650 42 in the first byte. Extensions, which are permitted, but are not defined in this specification, contain the  
1651 value 54 hex in the first byte. A list of defined Item Tags is in Section 9.1.3.1

##### 1652 9.1.1.2 Item Type

1653 An Item Type is a byte containing a coded value that indicates the data type of the data object. The  
1654 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

1655 **Table 192: Allowed Item Type Values**

1656 **9.1.1.3 Item Length**

1657 An Item Length is a 32-bit binary integer, transmitted big-endian, containing the number of bytes in the  
 1658 Item Value. The allowed values are:

1659

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

**Table 193: Allowed Item Length Values**

1660

1661 If the Item Type is Structure, then the Item Length is the total length of all of the sub-items contained in  
 1662 the structure, including any padding. If the Item Type is Integer, Enumeration, Text String, Byte String, or  
 1663 Interval, then the Item Length is the number of bytes excluding the padding bytes. Text Strings and Byte  
 1664 Strings SHALL be padded with the minimal number of bytes following the Item Value to obtain a multiple  
 1665 of eight bytes. Integers, Enumerations, and Intervals SHALL be padded with four bytes following the Item  
 1666 Value.

1667 **9.1.1.4 Item Value**

1668 The item value is a sequence of bytes containing the value of the data item, depending on the type:

- 1669 • Integers are encoded as four-byte long (32 bit) binary signed numbers in 2's complement  
 1670 notation, transmitted big-endian.
- 1671 • Long Integers are encoded as eight-byte long (64 bit) binary signed numbers in 2's complement  
 1672 notation, transmitted big-endian.
- 1673 • Big Integers are encoded as a sequence of eight-bit bytes, in two's complement notation,  
 1674 transmitted big-endian. If the length of the sequence is not a multiple of eight bytes, then Big  
 1675 Integers SHALL be padded with the minimal number of leading sign-extended bytes to make the  
 1676 length a multiple of eight bytes. These padding bytes are part of the Item Value and SHALL be  
 1677 counted in the Item Length.
- 1678 • Enumerations are encoded as four-byte long (32 bit) binary unsigned numbers transmitted big-  
 1679 endian. Extensions, which are permitted, but are not defined in this specification, contain the  
 1680 value 8 hex in the first nibble of the first byte.
- 1681 • Booleans are encoded as an eight-byte value that SHALL either contain the hex value  
 1682 0000000000000000, indicating the Boolean value *False*, or the hex value 0000000000000001,  
 1683 transmitted big-endian, indicating the Boolean value *True*.

- 1684 • Text Strings are sequences of bytes that encode character values according to the UTF-8  
1685 encoding standard. There SHALL NOT be null-termination at the end of such strings.
- 1686 • Byte Strings are sequences of bytes containing individual unspecified eight-bit binary values, and  
1687 are interpreted in the same sequence order.
- 1688 • Date-Time values are POSIX Time values encoded as Long Integers. POSIX Time, as described  
1689 in IEEE Standard 1003.1 [IEEE1003-1], is the number of seconds since the Epoch (1970 Jan 1,  
1690 00:00:00 UTC), not counting leap seconds.
- 1691 • Intervals are encoded as four-byte long (32 bit) binary unsigned numbers, transmitted big-endian.  
1692 They have a resolution of one second.
- 1693 • Structure Values are encoded as the concatenated encodings of the elements of the structure. All  
1694 structures defined in this specification SHALL have all of their fields encoded in the order in which  
1695 they appear in their respective structure descriptions.

## 1696 9.1.2 Examples

1697 These examples are assumed to be encoding a Protocol Object whose tag is 420020. The examples are  
1698 shown as a sequence of bytes in hexadecimal notation:

- 1699 • An Integer containing the decimal value 8:  
1700 42 00 20 | 02 | 00 00 00 04 | 00 00 00 08 00 00 00 00
- 1701 • A Long Integer containing the decimal value 123456789000000000:  
1702 42 00 20 | 03 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00
- 1703 • A Big Integer containing the decimal value 123456789000000000000000000000:  
1704 42 00 20 | 04 | 00 00 00 10 | 00 00 00 00 03 FD 35 EB 6B C2 DF 46 18 08  
1705 00 00
- 1706 • An Enumeration with value 255:  
1707 42 00 20 | 05 | 00 00 00 04 | 00 00 00 FF 00 00 00 00
- 1708 • A Boolean with the value *True*:  
1709 42 00 20 | 06 | 00 00 00 08 | 00 00 00 00 00 00 00 01
- 1710 • A Text String with the value "Hello World":  
1711 42 00 20 | 07 | 00 00 00 0B | 48 65 6C 6C 6F 20 57 6F 72 6C 64 00 00 00  
1712 00 00
- 1713 • A Byte String with the value { 0x01, 0x02, 0x03 }:  
1714 42 00 20 | 08 | 00 00 00 03 | 01 02 03 00 00 00 00 00
- 1715 • A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:  
1716 42 00 20 | 09 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8
- 1717 • An Interval, containing the value for 10 days:  
1718 42 00 20 | 0A | 00 00 00 04 | 00 0D 2F 00 00 00 00 00
- 1719 • A Structure containing an Enumeration, value 254, followed by an Integer, value 255, having tags  
1720 420004 and 420005 respectively:  
1721 42 00 20 | 01 | 00 00 00 20 | 42 00 04 | 05 | 00 00 00 04 | 00 00 00 FE  
1722 00 00 00 00 | 42 00 05 | 02 | 00 00 00 04 | 00 00 00 FF 00 00 00 00



1723 **9.1.3 Defined Values**

1724 This section specifies the values that are defined by this specification. In all cases where an extension  
 1725 mechanism is allowed, this extension mechanism is only able to be used for communication between  
 1726 parties that have pre-agreed understanding of the specific extensions.

1727 **9.1.3.1 Tags**

1728 The following table defines the tag values for the objects and primitive data values for the protocol  
 1729 messages.

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

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

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

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

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

**Table 194: Tag Values**

1731 **9.1.3.2 Enumerations**

1732 The following tables define the values for enumerated lists.

1733 **9.1.3.2.1 Credential Type Enumeration**

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

1734 **Table 195: Credential Type Enumeration**

1735 **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	8XXXXXXXX

1736 **Table 196: Key Compression Type Enumeration**

1737 **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	8XXXXXXXX

1738

**Table 197: Key Format Type Enumeration**

1739

### 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	8XXXXXXXX

1740

**Table 198: Wrapping Method Enumeration**

1741

### 9.1.3.2.5 Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV

1742

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

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

1743 **Table 199: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV**

1744 **9.1.3.2.6 Certificate Type Enumeration**

Certificate Type	
Name	Value
X.509	00000001
PGP	00000002
Extensions	8XXXXXXXX

1745 **Table 200: Certificate Type Enumeration**

1746 **9.1.3.2.7 Split Key Method Enumeration**

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

1747 **Table 201: Split Key Method Enumeration**



1748 **9.1.3.2.8 Secret Data Type Enumeration**

Secret Data Type	
Name	Value
Password	00000001
Seed	00000002
Extensions	8XXXXXXXX

1749 **Table 202: Secret Data Type Enumeration**

1750 **9.1.3.2.9 Opaque Data Type Enumeration**

Opaque Data Type	
Name	Value
Extensions	8XXXXXXXX

1751 **Table 203: Opaque Data Type Enumeration**

1752 **9.1.3.2.10 Name Type Enumeration**

Name Type	
Name	Value
Uninterpreted Text String	00000001
URI	00000002
Extensions	8XXXXXXXX

1753 **Table 204: Name Type Enumeration**

1754 **9.1.3.2.11 Object Type Enumeration**

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

1755 **Table 205: Object Type Enumeration**

1756 **9.1.3.2.12 Cryptographic Algorithm Enumeration**

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

1757 **Table 206: Cryptographic Algorithm Enumeration**

1758 **9.1.3.2.13 Block Cipher Mode Enumeration**

Block Cipher Mode	
Name	Value
CBC	00000001
ECB	00000002
PCBC	00000003
CFB	00000004
OFB	00000005
CTR	00000006
CMAC	00000007
CCM	00000008
GCM	00000009
CBC-MAC	0000000A
XTS	0000000B
AESKeyWrapPadding	0000000C
NISTKeyWrap	0000000D
X9.102 AESKW	0000000E
X9.102 TDKW	0000000F
X9.102 AKW1	00000010
X9.102 AKW2	00000011
Extensions	8XXXXXXXX

1759 **Table 207: Block Cipher Mode Enumeration**

1760 **9.1.3.2.14 Padding Method Enumeration**

Padding Method	
Name	Value
None	00000001
OAEP	00000002
PKCS5	00000003
SSL3	00000004
Zeros	00000005
ANSI X9.23	00000006
ISO 10126	00000007
PKCS1 v1.5	00000008
X9.31	00000009
PSS	0000000A
Extensions	8XXXXXXXX

1761 **Table 208: Padding Method Enumeration**

1762 **9.1.3.2.15 Hashing Algorithm Enumeration**

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

1763 **Table 209: Hashing Algorithm Enumeration**

1764 **9.1.3.2.16 Role Type Enumeration**

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

**Table 210: Role Type Enumeration**

1765  
 1766 Note that while the set and definitions of role types are chosen to match TR-31 there is no necessity to  
 1767 match binary representations.

1768 **9.1.3.2.17 State Enumeration**

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

Extensions	8XXXXXXXX
------------	-----------

1769

**Table 211: State Enumeration**

1770

**9.1.3.2.18 Revocation Reason Code Enumeration**

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

1771

**Table 212: Revocation Reason Code Enumeration**

1772

**9.1.3.2.19 Link Type Enumeration**

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

1773

**Table 213: Link Type Enumeration**

1774

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

1775 **9.1.3.2.20 Derivation Method Enumeration**

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

1776 **Table 214: Derivation Method Enumeration**

1777 **9.1.3.2.21 Certificate Request Type Enumeration**

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

1778 **Table 215: Certificate Request Type Enumeration**

1779 **9.1.3.2.22 Validity Indicator Enumeration**

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

1780 **Table 216: Validity Indicator Enumeration**

1781 **9.1.3.2.23 Query Function Enumeration**

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

Query Application Namespaces	00000004
Extensions	8XXXXXXXX

1782

**Table 217: Query Function Enumeration**

1783

**9.1.3.2.24 Cancellation Result Enumeration**

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

1784

**Table 218: Cancellation Result Enumeration**

1785

**9.1.3.2.25 Put Function Enumeration**

Put Function	
Name	Value
New	00000001
Replace	00000002
Extensions	8XXXXXXXX

1786

**Table 219: Put Function Enumeration**



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

Table 220: Operation Enumeration

1789 **9.1.3.2.27 Result Status Enumeration**

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

1790 **Table 221: Result Status Enumeration**

1791 **9.1.3.2.28 Result Reason Enumeration**

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

1792 **Table 222: Result Reason Enumeration**

1793 **9.1.3.2.29 Batch Error Continuation Enumeration**

Batch Error Continuation	
Name	Value
Continue	00000001
Stop	00000002
Undo	00000003

Extensions	8xxxxxxxx
------------	-----------

1794

**Table 223: Batch Error Continuation Enumeration**

1795 **9.1.3.3 Bit Masks**

1796 **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

1797

**Table 224: Cryptographic Usage Mask**

1798 This list takes into consideration values which MAY appear in the Key Usage extension in an X.509  
 1799 certificate.

1800 **9.1.3.3.2 Storage Status Mask**

Storage Status Mask	
Name	Value
On-line storage	00000001
Archival storage	00000002
Extensions	xxxxxxx0

1801 **Table 225: Storage Status Mask**

1802 **9.2 XML Encoding**

1803 An XML Encoding has not yet been defined.

---

1804 **10Transport**

1805 A KMIP Server SHALL establish and maintain channel confidentiality and integrity, and prove server  
1806 authenticity for KMIP messaging.

1807 If a KMIP Server uses TCP/IP for KMIP messaging, then it SHALL support SSL v3.1/TLS v1.0 or later and  
1808 may support other protocols as specified in **[KMIP-Prof]**.

---

1809 **11 Error Handling**

1810 This section details the specific Result Reasons that SHALL be returned for errors detected.

1811 **11.1 General**

1812 These errors MAY occur when any protocol message is received by the server.

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

1813

**Table 226: General Errors**

1814 **11.2 Create**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
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

1815 **Table 227: Create Errors**

1816 **11.3 Create Key Pair**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
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

1817 **Table 228: Create Key Pair Errors**

1818 **11.4 Register**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
Object Type is not recognized	Operation Failed	Invalid Field
Object Type does not match type of cryptographic object provided	Operation Failed	Invalid Field
Templates that do not exist are given in request	Operation Failed	Item Not Found
Incorrect attribute value(s) specified	Operation Failed	Invalid Field
Trying to register a new object with the same Name attribute value as an existing object	Operation Failed	Invalid Field
Trying to set more instances than the server supports of an attribute that MAY have multiple instances	Operation Failed	Index Out of Bounds
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Template object is archived	Operation Failed	Object Archived

1819 **Table 229: Register Errors**

1820 **11.5 Re-key**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
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

1821 **Table 230: Re-key Errors**



## 11.6 Derive Key

Error Definition	Result Status	Result Reason
One or more of the objects specified do not exist	Operation Failed	Item Not Found
One or more of the objects specified are not of the correct type	Operation Failed	Invalid Field
Templates that do not exist are given in request	Operation Failed	Item Not Found
Invalid Derivation Method	Operation Failed	Invalid Field
Invalid Derivation Parameters	Operation Failed	Invalid Field
Ambiguous derivation data provided both with Derivation Data and Secret Data object.	Operation Failed	Invalid Message
Incorrect attribute value(s) specified	Operation Failed	Invalid Field
One or more of the specified objects are not able to be used to derive a new key	Operation Failed	Invalid Field
Trying to derive a new key with the same Name attribute value as an existing object	Operation Failed	Invalid Field
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
One or more of the objects is archived	Operation Failed	Object Archived

Table 231: Derive Key Errors-

## 11.7 Certify

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object specified is not able to be certified	Operation Failed	Permission Denied
The Certificate Request does not contain a signed certificate request of the specified Certificate Request Type	Operation Failed	Invalid Field
Server does not support operation	Operation Failed	Operation Not Supported
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported

Object is archived	Operation Failed	Object Archived
--------------------	------------------	-----------------

1825

**Table 232: Certify Errors**

1826

## 11.8 Re-certify

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object specified is not able to be certified	Operation Failed	Permission Denied
The Certificate Request does not contain a signed certificate request of the specified Certificate Request Type	Operation Failed	Invalid Field
Server does not support operation	Operation Failed	Operation Not Supported
Offset field is not permitted to be specified at the same time as any of the Activation Date or Deactivation Date attributes	Operation Failed	Invalid Message
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived

1827

**Table 233: Re-certify Errors**

1828

## 11.9 Locate

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

1829

**Table 234: Locate Errors**

1830

## 11.10 Check

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

1831

**Table 235: Check Errors**

1832 **11.11 Get**

Error Definition	Result Status	Result Reason
Object does not exist	Operation Failed	Item Not Found
Wrapping key does not exist	Operation Failed	Item Not Found
Object with Wrapping Key ID exists, but it is not a key	Operation Failed	Illegal Operation
Object with Wrapping Key ID exists, but it is not able to be used for wrapping	Operation Failed	Permission Denied
Object with MAC/Signature Key ID exists, but it is not a key	Operation Failed	Illegal Operation
Object with MAC/Signature Key ID exists, but it 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

1833 **Table 236: Get Errors**

1834 **11.12 Get Attributes**

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

1835 **Table 237: Get Attributes Errors**

1836 **11.13 Get Attribute List**

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

Object is archived	Operation Failed	Object Archived
--------------------	------------------	-----------------

1837

**Table 238: Get Attribute List Errors**

1838

**11.14 Add Attribute**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
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

1839

**Table 239: Add Attribute Errors**

1840

**11.15 Modify Attribute**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
A specified attribute does not exist (i.e., it needs to first be added)	Operation Failed	Invalid Field
An Attribute Index is specified, but no matching instance exists.	Operation Failed	Item Not Found
The specified attribute is read-only	Operation Failed	Permission Denied
Trying to set the Name attribute value to a value already used by another object	Operation Failed	Illegal Operation
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted	Operation Failed	Application Namespace Not Supported

from the client request		
Object is archived	Operation Failed	Object Archived

1841

**Table 240: Modify Attribute Errors**1842 **11.16 Delete Attribute**

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

1843

**Table 241: Delete Attribute Errors**1844 **11.17 Obtain Lease**

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
The server determines that a new lease is not permitted to be issued for the specified cryptographic object	Operation Failed	Permission Denied
Object is archived	Operation Failed	Object Archived

1845

**Table 242: Obtain Lease Errors**1846 **11.18 Get Usage Allocation**

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object has no Usage Limits attribute, or the object is not able to be used for applying cryptographic protection	Operation Failed	Illegal Operation
Both Usage Limits Byte Count and Usage Limits Object Count fields are specified	Operation Failed	Invalid Message
Neither the Byte Count or Object Count is specified	Operation Failed	Invalid Message

A usage type (Byte Count or Object Count) is specified in the request, but the usage allocation for the object MAY only be given for the other type	Operation Failed	Operation Not Supported
Object is archived	Operation Failed	Object Archived

1847

**Table 243: Get Usage Allocation Errors**

1848

**11.19 Activate**

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

1849

**Table 244: Activate Errors**

1850

**11.20 Revoke**

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Revocation Reason is not recognized	Operation Failed	Invalid Field
Unique Identifier specifies a template or other object that is not able to be revoked	Operation Failed	Illegal Operation
Object is archived	Operation Failed	Object Archived

1851

**Table 245: Revoke Errors**

1852

**11.21 Destroy**

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

1853

**Table 246: Destroy Errors**

1854 **11.22 Archive**

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

1855 **Table 247: Archive Errors**

1856 **11.23 Recover**

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

1857 **Table 248: Recover Errors**

1858 **11.24 Validate**

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

1859 **Table 249: Validate Errors**

1860 **11.25 Query**

1861 N/A

1862 **11.26 Cancel**

1863 N/A

1864 **11.27 Poll**

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

1865 **Table 250: Poll Errors**

1866 **11.28 Batch Items**

1867 These errors MAY occur when a protocol message with one or more batch items is processed by the  
1868 server. If a message with one or more batch items was parsed correctly, then the response message  
1869 SHOULD include response(s) to the batch item(s) in the request according to the table below.

1870

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

1871

**Table 251: Batch Items Errors**



---

## 1872 12 Implementation Conformance

1873 The intention of the baseline conformance profile is for the minimal KMIP Server to support the  
1874 mechanics of communication and to support a limited set of commands, such as query. The minimal  
1875 KMIP Server would not need to support any particular algorithm – this would be the work of additional  
1876 profiles.

1877 An implementation is a conforming KMIP Server if the implementation meets the conditions in Section  
1878 12.1 .

1879 An implementation SHALL be a conforming KMIP Server.

1880 If an implementation claims support for a particular clause, then the implementation SHALL conform to all  
1881 normative statements within that clause and any subclauses to that clause.

### 1882 12.1 Conformance clauses for a KMIP Server

1883 An implementation conforms to this specification as a KMIP Server if it meets the following conditions:

- 1884 1. Supports the following objects:
  - 1885 a. Attribute (see 2.1.1 )
  - 1886 b. Credential (see 2.1.2 )
  - 1887 c. Key Block (see 2.1.3 )
  - 1888 d. Key Value (see 2.1.4 )
  - 1889 e. Template-Attribute Structure (see 2.1.8 )
- 1890 2. Supports the following attributes:
  - 1891 a. Unique Identifier (see 3.1 )
  - 1892 b. Name (see 3.2 )
  - 1893 c. Object Type (see 3.3 )
  - 1894 d. Cryptographic Algorithm (see 3.4 )
  - 1895 e. Cryptographic Length (see 3.5 )
  - 1896 f. Cryptographic Parameters (see 3.6 )
  - 1897 g. Digest (see 3.12 )
  - 1898 h. Default Operation Policy (see 3.13.2 )
  - 1899 i. Cryptographic Usage Mask (see 3.14 )
  - 1900 j. State (see 3.17 )
  - 1901 k. Initial Date (see 3.18 )
  - 1902 l. Activation Date (see 3.19 )
  - 1903 m. Deactivation Date (see 3.22 )
  - 1904 n. Destroy Date (see 3.23 )
  - 1905 o. Compromise Occurrence Date (see 3.24 )
  - 1906 p. Compromise Date (see 3.25 )
  - 1907 q. Revocation Reason (see 3.26 )
  - 1908 r. Archive Date (see 3.27 )
  - 1909 s. Last Change Date (see 3.32 )
- 1910 3. Supports the following client-to-server operations:
  - 1911 a. Locate (see 4.8 )

- 1912            b. Check (see 4.9 )
- 1913            c. Get (see 4.10 )
- 1914            d. Get Attribute (see 4.11 )
- 1915            e. Get Attribute List (see 4.12 )
- 1916            f. Add Attribute (see 4.13 )
- 1917            g. Modify Attribute (see 4.14 )
- 1918            h. Delete Attribute (see 4.15 )
- 1919            i. Activate (see 4.18 )
- 1920            j. Revoke (see 4.19 )
- 1921            k. Destroy (see 4.20 )
- 1922            l. Query (see 4.24 )
- 1923            4. Supports the following message contents:
- 1924                a. Protocol Version (see 6.1 )
- 1925                b. Operation (see 6.2 )
- 1926                c. Maximum Response Size (see 6.3 )
- 1927                d. Unique Batch Item ID (see 6.4 )
- 1928                e. Time Stamp (see 6.5 )
- 1929                f. Asynchronous Indicator (see 6.7 )
- 1930                g. Result Status (see 6.9 )
- 1931                h. Result Reason (see 6.10 )
- 1932                i. Result Message (see 6.11 )
- 1933                j. Batch Order Option (see 6.12 )
- 1934                k. Batch Error Continuation Option (see 6.13 )
- 1935                l. Batch Count (see 6.14 )
- 1936                m. Batch Item (see 6.15 )
- 1937            5. Supports Message Format (see 7 )
- 1938            6. Supports Authentication (see 8 )
- 1939            7. Supports the TTLV encoding (see 9.1 )
- 1940            8. Supports the transport requirements (see 10 )
- 1941            9. Supports Error Handling (see 11 ) for any supported object, attribute, or operation
- 1942            10. Optionally supports any clause within this specification that is not listed above
- 1943            11. Optionally supports extensions outside the scope of this standard (e.g., vendor extensions,
- 1944                conformance profiles) that do not contradict any requirements within this standard
- 1945            12. Supports at least one of the profiles defined in the KMIP Profiles Specification **[KMIP-Prof]**.
- 1946

1947

## A. Attribute Cross-reference

1948

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

1949

This table is not normative.

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

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

1950

**Table 252: Attribute Cross-reference**

1951

## B. Tag Cross-reference

1952

This table is not normative.

Object	Defined	Type	Notes
Activation Date	3.19	Date-Time	
Application Data	3.30	Text String	
Application Namespace	3.30	Text String	
Application Specific Information	3.30	Structure	
Archive Date	3.27	Date-Time	
Asynchronous Correlation Value	6.8	Byte String	
Asynchronous Indicator	6.7	Boolean	
Attribute	2.1.1	Structure	
Attribute Index	2.1.1	Integer	
Attribute Name	2.1.1	Text String	
Attribute Value	2.1.1	*	type varies
Authentication	6.6	Structure	
Batch Count	6.14	Integer	
Batch Error Continuation Option	6.13 , 9.1.3.2.29	Enumeration	
Batch Item	6.15	Structure	
Batch Order Option	6.12	Boolean	
Block Cipher Mode	3.6 , 9.1.3.2.13	Enumeration	
Cancellation Result	4.25 , 9.1.3.2.24	Enumeration	
Certificate	2.2.1	Structure	
Certificate Identifier	3.9	Structure	
Certificate Issuer	3.9	Structure	
Certificate Issuer Alternative Name	3.11	Text String	
Certificate Issuer Distinguished Name	3.11	Text String	
Certificate Request	4.6 , 4.7	Byte String	
Certificate Request Type	4.6 , 4.7 , 9.1.3.2.21	Enumeration	
Certificate Subject	3.10	Structure	
Certificate Subject Alternative Name	3.10	Text String	
Certificate Subject Distinguished Name	3.10	Text String	
Certificate Type	2.2.1 , 3.8 , 9.1.3.2.6	Enumeration	
Certificate Value	2.2.1	Byte String	
Common Template-Attribute	2.1.8	Structure	
Compromise Occurrence Date	3.24	Date-Time	
Compromise Date	3.25	Date-Time	
Contact Information	3.31	Text String	

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

Object	Defined	Type	Notes
Last Change Date	3.32	Date-Time	
Lease Time	3.15	Interval	
Link	3.29	Structure	
Link Type	3.29 , 9.1.3.2.19	Enumeration	
Linked Object Identifier	3.29	Text String	
MAC/Signature	2.1.5	Byte String	
MAC/Signature Key Information	2.1.5	Text String	
Maximum Items	4.8	Integer	
Maximum Response Size	6.3	Integer	
Message Extension	6.16	Structure	
Modulus	2.1.7	Big Integer	
Name	3.2	Structure	
Name Type	3.2 , 9.1.3.2.10	Enumeration	
Name Value	3.2	Text String	
Object Group	3.28	Text String	
Object Type	3.3 , 9.1.3.2.11	Enumeration	
Offset	4.4 , 4.7	Interval	
Opaque Data Type	2.2.8 , 9.1.3.2.9	Enumeration	
Opaque Data Value	2.2.8	Byte String	
Opaque Object	2.2.8	Structure	
Operation	6.2 , 9.1.3.2.26	Enumeration	
Operation Policy Name	3.13	Text String	
P	2.1.7	Big Integer	
Padding Method	3.6 , 9.1.3.2.14	Enumeration	
Prime Exponent P	2.1.7	Big Integer	
Prime Exponent Q	2.1.7	Big Integer	
Prime Field Size	2.2.5	Big Integer	
Private Exponent	2.1.7	Big Integer	
Private Key	2.2.4	Structure	
Private Key Template-Attribute	2.1.8	Structure	
Private Key Unique Identifier	4.2	Text String	
Process Start Date	3.20	Date-Time	
Protect Stop Date	3.21	Date-Time	
Protocol Version	6.1	Structure	
Protocol Version Major	6.1	Integer	
Protocol Version Minor	6.1	Integer	
Public Exponent	2.1.7	Big Integer	
Public Key	2.2.3	Structure	

Object	Defined	Type	Notes
Public Key Template-Attribute	2.1.8	Structure	
Public Key Unique Identifier	4.2	Text String	
Put Function	5.2 , 9.1.3.2.25	Enumeration	
Q	2.1.7	Big Integer	
Q String	2.1.7	Byte String	
Qlength	3.7	Integer	
Query Function	4.24 , 9.1.3.2.23	Enumeration	
Recommended Curve	2.1.7 , 3.7 , 9.1.3.2.5	Enumeration	
Replaced Unique Identifier	5.2	Text String	
Request Header	7.2 , 7.3	Structure	
Request Message	7.1	Structure	
Request Payload	4 , 5 , 7.2 , 7.3	Structure	
Response Header	7.2 , 7.3	Structure	
Response Message	7.1	Structure	
Response Payload	4 , 7.2 , 7.3	Structure	
Result Message	6.11	Text String	
Result Reason	6.10 , 9.1.3.2.28	Enumeration	
Result Status	6.9 , 9.1.3.2.27	Enumeration	
Revocation Message	3.26	Text String	
Revocation Reason	3.26	Structure	
Revocation Reason Code	3.26 , 9.1.3.2.18	Enumeration	
Role Type	3.6 , 9.1.3.2.16	Enumeration	
Salt	4.5	Byte String	
Secret Data	2.2.7	Structure	
Secret Data Type	2.2.7 , 9.1.3.2.8	Enumeration	
Serial Number	3.9	Text String	
Server Information	4.24	Structure	contents vendor-specific
Split Key	2.2.5	Structure	
Split Key Method	2.2.5 , 9.1.3.2.7	Enumeration	
Split Key Parts	2.2.5	Integer	
Split Key Threshold	2.2.5	Integer	
State	3.17 , 9.1.3.2.17	Enumeration	
Storage Status Mask	4.8 , 9.1.3.3.2	Integer	Bit mask
Symmetric Key	2.2.2	Structure	
Template	2.2.6	Structure	
Template-Attribute	2.1.8	Structure	
Time Stamp	6.5	Date-Time	
Transparent*	2.1.7	Structure	



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

1953

**Table 253: Tag Cross-reference**

1954

1955

## C. Operation and Object Cross-reference

1956

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

1957

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

1958

**Table 254: Operation and Object Cross-reference**

## 1959 **D. Acronyms**

1960 The following abbreviations and acronyms are used in this document:

- 1961 3DES - Triple Data Encryption Standard specified in ANSI X9.52
- 1962 AES - Advanced Encryption Standard specified in FIPS 197
- 1963 ASN.1 - Abstract Syntax Notation One specified in ITU-T X.680
- 1964 BDK - Base Derivation Key specified in ANSI X9 TR-31
- 1965 CA - Certification Authority
- 1966 CBC - Cipher Block Chaining
- 1967 CCM - Counter with CBC-MAC specified in NIST SP 800-38C
- 1968 CFB - Cipher Feedback specified in NIST SP 800-38A
- 1969 CMAC - Cipher-based MAC specified in NIST SP 800-38B
- 1970 CMC - Certificate Management Messages over CMS specified in RFC 5275
- 1971 CMP - Certificate Management Protocol specified in RFC 4210
- 1972 CPU - Central Processing Unit
- 1973 CRL - Certificate Revocation List specified in RFC 5280
- 1974 CRMF - Certificate Request Message Format specified in RFC 4211
- 1975 CRT - Chinese Remainder Theorem
- 1976 CTR - Counter specified in NIST SP 800-38A
- 1977 CVK - Card Verification Key specified in ANSI X9 TR-31
- 1978 DEK - Data Encryption Key
- 1979 DER - Distinguished Encoding Rules specified in ITU-T X.690
- 1980 DES - Data Encryption Standard specified in FIPS 46-3
- 1981 DH - Diffie-Hellman specified in ANSI X9.42
- 1982 DNS - Domain Name Server
- 1983 DSA - Digital Signature Algorithm specified in FIPS 186-3
- 1984 DSKPP - Dynamic Symmetric Key Provisioning Protocol
- 1985 ECB - Electronic Code Book
- 1986 ECDH - Elliptic Curve Diffie-Hellman specified in ANSI X9.63 and NIST SP 800-56A
- 1987 ECDSA - Elliptic Curve Digital Signature Algorithm specified in ANSX9.62
- 1988 ECMQV - Elliptic Curve Menezes Qu Vanstone specified in ANSI X9.63 and NIST SP 800-56A
- 1989 FIPS - Federal Information Processing Standard
- 1990 GCM - Galois/Counter Mode specified in NIST SP 800-38D
- 1991 GF - Galois field (or finite field)
- 1992 HMAC - Keyed-Hash Message Authentication Code specified in FIPS 198-1 and RFC 2104
- 1993 HTTP - Hyper Text Transfer Protocol
- 1994 HTTP(S) - Hyper Text Transfer Protocol (Secure socket)
- 1995 IEEE - Institute of Electrical and Electronics Engineers

1996	IETF	- Internet Engineering Task Force
1997	IP	- Internet Protocol
1998	IPsec	- Internet Protocol Security
1999	IV	- Initialization Vector
2000	KEK	- Key Encryption Key
2001	KMIP	- Key Management Interoperability Protocol
2002	MAC	- Message Authentication Code
2003	MKAC	- EMV/chip card Master Key: Application Cryptograms specified in ANSI X9 TR-31
2004	MKCP	- EMV/chip card Master Key: Card Personalization specified in ANSI X9 TR-31
2005	MKDAC	- EMV/chip card Master Key: Data Authentication Code specified in ANSI X9 TR-31
2006	MKDN	- EMV/chip card Master Key: Dynamic Numbers specified in ANSI X9 TR-31
2007	MKOTH	- EMV/chip card Master Key: Other specified in ANSI X9 TR-31
2008	MKSMC	- EMV/chip card Master Key: Secure Messaging for Confidentiality specified in X9 TR-31
2009	MKSMI	- EMV/chip card Master Key: Secure Messaging for Integrity specified in ANSI X9 TR-31
2010	MD2	- Message Digest 2 Algorithm specified in RFC 1319
2011	MD4	- Message Digest 4 Algorithm specified in RFC 1320
2012	MD5	- Message Digest 5 Algorithm specified in RFC 1321
2013	NIST	- National Institute of Standards and Technology
2014	OAEP	- Optimal Asymmetric Encryption Padding specified in PKCS#1
2015	OFB	- Output Feedback specified in NIST SP 800-38A
2016	PBKDF2	- Password-Based Key Derivation Function 2 specified in RFC 2898
2017	PCBC	- Propagating Cipher Block Chaining
2018	PEM	- Privacy Enhanced Mail specified in RFC 1421
2019	PGP	- Pretty Good Privacy specified in RFC 1991
2020	PKCS	- Public-Key Cryptography Standards
2021	PKCS#1	- RSA Cryptography Specification Version 2.1 specified in RFC 3447
2022	PKCS#5	- Password-Based Cryptography Specification Version 2 specified in RFC 2898
2023	PKCS#8	- Private-Key Information Syntax Specification Version 1.2 specified in RFC 5208
2024	PKCS#10	- Certification Request Syntax Specification Version 1.7 specified in RFC 2986
2025	POSIX	- Portable Operating System Interface
2026	RFC	- Request for Comments documents of IETF
2027	RSA	- Rivest, Shamir, Adelman (an algorithm)
2028	SCEP	- Simple Certificate Enrollment Protocol
2029	SHA	- Secure Hash Algorithm specified in FIPS 180-2
2030	SP	- Special Publication
2031	SSL/TLS	- Secure Sockets Layer/Transport Layer Security
2032	S/MIME	- Secure/Multipurpose Internet Mail Extensions
2033	TDEA	- see 3DES

2034	TCP	- Transport Control Protocol
2035	TTLV	- Tag, Type, Length, Value
2036	URI	- Uniform Resource Identifier
2037	UTC	- Universal Time Coordinated
2038	UTF	- Universal Transformation Format 8-bit specified in RFC 3629
2039	XKMS	- XML Key Management Specification
2040	XML	- Extensible Markup Language
2041	XTS	- XEX Tweakable Block Cipher with Ciphertext Stealing specified in NIST SP 800-38E
2042	X.509	- Public Key Certificate specified in RFC 5280
2043	ZPK	- PIN Block Encryption Key specified in ANSI X9 TR-31

---

## 2044 E. List of Figures and Tables

2045	Figure 1: Cryptographic Object States and Transitions .....	44
2046		
2047	Table 1: Attribute Object Structure.....	14
2048	Table 2: Credential Object Structure.....	15
2049	Table 3: Key Block Object Structure .....	16
2050	Table 4: Key Value Object Structure.....	17
2051	Table 5: Key Wrapping Data Object Structure.....	18
2052	Table 6: Encryption Key Information Object Structure.....	18
2053	Table 7: MAC/Signature Key Information Object Structure .....	18
2054	Table 8: Key Wrapping Specification Object Structure .....	19
2055	Table 9: Key Material Object Structure for Transparent Symmetric Keys .....	19
2056	Table 10: Key Material Object Structure for Transparent DSA Private Keys .....	19
2057	Table 11: Key Material Object Structure for Transparent DSA Public Keys .....	20
2058	Table 12: Key Material Object Structure for Transparent RSA Private Keys .....	20
2059	Table 13: Key Material Object Structure for Transparent RSA Public Keys .....	21
2060	Table 14: Key Material Object Structure for Transparent DH Private Keys.....	21
2061	Table 15: Key Material Object Structure for Transparent DH Public Keys .....	21
2062	Table 16: Key Material Object Structure for Transparent ECDSA Private Keys .....	22
2063	Table 17: Key Material Object Structure for Transparent ECDSA Public Keys.....	22
2064	Table 18: Key Material Object Structure for Transparent ECDH Private Keys.....	22
2065	Table 19: Key Material Object Structure for Transparent ECDH Public Keys .....	22
2066	Table 20: Key Material Object Structure for Transparent ECMQV Private Keys.....	23
2067	Table 21: Key Material Object Structure for Transparent ECMQV Public Keys .....	23
2068	Table 22: Template-Attribute Object Structure .....	23
2069	Table 23: Certificate Object Structure .....	24
2070	Table 24: Symmetric Key Object Structure.....	24
2071	Table 25: Public Key Object Structure .....	24
2072	Table 26: Private Key Object Structure.....	24
2073	Table 27: Split Key Object Structure .....	25
2074	Table 28: Template Object Structure .....	26
2075	Table 29: Secret Data Object Structure .....	27
2076	Table 30: Opaque Object Structure .....	27
2077	Table 31: Attribute Rules.....	28
2078	Table 32: Unique Identifier Attribute .....	29
2079	Table 33: Unique Identifier Attribute Rules .....	29
2080	Table 34: Name Attribute Structure .....	29
2081	Table 35: Name Attribute Rules .....	30
2082	Table 36: Object Type Attribute .....	30
2083	Table 37: Object Type Attribute Rules .....	30

2084	Table 38: Cryptographic Algorithm Attribute .....	30
2085	Table 39: Cryptographic Algorithm Attribute Rules .....	31
2086	Table 40: Cryptographic Length Attribute .....	31
2087	Table 41: Cryptographic Length Attribute Rules .....	31
2088	Table 42: Cryptographic Parameters Attribute Structure .....	32
2089	Table 43: Cryptographic Parameters Attribute Rules .....	32
2090	Table 44: Role Types .....	33
2091	Table 45: Cryptographic Domain Parameters Attribute Structure .....	34
2092	Table 46: Cryptographic Domain Parameters Attribute Rules .....	34
2093	Table 47: Certificate Type Attribute .....	34
2094	Table 48: Certificate Type Attribute Rules .....	34
2095	Table 49: Certificate Identifier Attribute Structure .....	35
2096	Table 50: Certificate Identifier Attribute Rules .....	35
2097	Table 51: Certificate Subject Attribute Structure .....	35
2098	Table 52: Certificate Subject Attribute Rules .....	36
2099	Table 53: Certificate Issuer Attribute Structure .....	36
2100	Table 54: Certificate Issuer Attribute Rules .....	36
2101	Table 55: Digest Attribute Structure .....	37
2102	Table 56: Digest Attribute Rules .....	37
2103	Table 57: Operation Policy Name Attribute .....	37
2104	Table 58: Operation Policy Name Attribute Rules .....	38
2105	Table 59: Default Operation Policy for Secret Objects .....	39
2106	Table 60: Default Operation Policy for Certificates and Public Key Objects .....	39
2107	Table 61: Default Operation Policy for Private Template Objects .....	40
2108	Table 62: Default Operation Policy for Public Template Objects .....	40
2109	Table 63: X.509 Key Usage to Cryptographic Usage Mask Mapping .....	41
2110	Table 64: Cryptographic Usage Mask Attribute .....	41
2111	Table 65: Cryptographic Usage Mask Attribute Rules .....	42
2112	Table 66: Lease Time Attribute .....	42
2113	Table 67: Lease Time Attribute Rules .....	42
2114	Table 68: Usage Limits Attribute Structure .....	43
2115	Table 69: Usage Limits Attribute Rules .....	44
2116	Table 70: State Attribute .....	46
2117	Table 71: State Attribute Rules .....	46
2118	Table 72: Initial Date Attribute .....	46
2119	Table 73: Initial Date Attribute Rules .....	46
2120	Table 74: Activation Date Attribute .....	47
2121	Table 75: Activation Date Attribute Rules .....	47
2122	Table 76: Process Start Date Attribute .....	47
2123	Table 77: Process Start Date Attribute Rules .....	47
2124	Table 78: Protect Stop Date Attribute .....	48
2125	Table 79: Protect Stop Date Attribute Rules .....	48

2126	Table 80: Deactivation Date Attribute .....	48
2127	Table 81: Deactivation Date Attribute Rules .....	48
2128	Table 82: Destroy Date Attribute.....	49
2129	Table 83: Destroy Date Attribute Rules .....	49
2130	Table 84: Compromise Occurrence Date Attribute .....	49
2131	Table 85: Compromise Occurrence Date Attribute Rules.....	49
2132	Table 86: Compromise Date Attribute.....	50
2133	Table 87: Compromise Date Attribute Rules .....	50
2134	Table 88: Revocation Reason Attribute Structure .....	50
2135	Table 89: Revocation Reason Attribute Rules .....	50
2136	Table 90: Archive Date Attribute .....	51
2137	Table 91: Archive Date Attribute Rules.....	51
2138	Table 92: Object Group Attribute .....	51
2139	Table 93: Object Group Attribute Rules .....	51
2140	Table 94: Link Attribute Structure .....	52
2141	Table 95: Link Attribute Structure Rules .....	53
2142	Table 96: Application Specific Information Attribute .....	53
2143	Table 97: Application Specific Information Attribute Rules .....	53
2144	Table 98: Contact Information Attribute .....	53
2145	Table 99: Contact Information Attribute Rules.....	54
2146	Table 100: Last Change Date Attribute.....	54
2147	Table 101: Last Change Date Attribute Rules .....	54
2148	Table 102 Custom Attribute .....	55
2149	Table 103: Custom Attribute Rules .....	55
2150	Table 104: Create Request Payload.....	57
2151	Table 105: Create Response Payload .....	57
2152	Table 106: Create Attribute Requirements .....	57
2153	Table 107: Create Key Pair Request Payload .....	58
2154	Table 108: Create Key Pair Response Payload .....	58
2155	Table 109: Create Key Pair Attribute Requirements.....	59
2156	Table 110: Register Request Payload .....	59
2157	Table 111: Register Response Payload .....	60
2158	Table 112: Register Attribute Requirements.....	60
2159	Table 113: Computing New Dates from Offset during Re-key.....	61
2160	Table 114: Re-key Attribute Requirements.....	61
2161	Table 115: Re-key Request Payload .....	62
2162	Table 116: Re-key Response Payload .....	62
2163	Table 117: Derive Key Request Payload .....	63
2164	Table 118: Derive Key Response Payload .....	63
2165	Table 119: Derivation Parameters Structure (Except PBKDF2).....	64
2166	Table 120: PBKDF2 Derivation Parameters Structure .....	64
2167	Table 121: Certify Request Payload .....	65



2168	Table 122: Certify Response Payload .....	65
2169	Table 123: Computing New Dates from Offset during Re-certify.....	66
2170	Table 124: Re-certify Attribute Requirements.....	66
2171	Table 125: Re-certify Request Payload .....	67
2172	Table 126: Re-certify Response Payload .....	67
2173	Table 127: Locate Request Payload.....	68
2174	Table 128: Locate Response Payload .....	68
2175	Table 129: Check Request Payload .....	70
2176	Table 130: Check Response Payload.....	70
2177	Table 131: Get Request Payload .....	71
2178	Table 132: Get Response Payload .....	71
2179	Table 133: Get Attributes Request Payload.....	72
2180	Table 134: Get Attributes Response Payload.....	72
2181	Table 135: Get Attribute List Request Payload.....	72
2182	Table 136: Get Attribute List Response Payload .....	72
2183	Table 137: Add Attribute Request Payload.....	73
2184	Table 138: Add Attribute Response Payload.....	73
2185	Table 139: Modify Attribute Request Payload.....	73
2186	Table 140: Modify Attribute Response Payload.....	73
2187	Table 141: Delete Attribute Request Payload.....	74
2188	Table 142: Delete Attribute Response Payload .....	74
2189	Table 143: Obtain Lease Request Payload .....	74
2190	Table 144: Obtain Lease Response Payload .....	75
2191	Table 145: Get Usage Allocation Request Payload.....	75
2192	Table 146: Get Usage Allocation Response Payload.....	76
2193	Table 147: Activate Request Payload.....	76
2194	Table 148: Activate Response Payload .....	76
2195	Table 149: Revoke Request Payload .....	76
2196	Table 150: Revoke Response Payload.....	76
2197	Table 151: Destroy Request Payload .....	77
2198	Table 152: Destroy Response Payload .....	77
2199	Table 153: Archive Request Payload.....	77
2200	Table 154: Archive Response Payload.....	77
2201	Table 155: Recover Request Payload .....	78
2202	Table 156: Recover Response Payload .....	78
2203	Table 157: Validate Request Payload.....	78
2204	Table 158: Validate Response Payload.....	78
2205	Table 159: Query Request Payload.....	79
2206	Table 160: Query Response Payload .....	80
2207	Table 161: Cancel Request Payload .....	80
2208	Table 162: Cancel Response Payload.....	80
2209	Table 163: Poll Request Payload.....	81

2210	Table 164: Notify Message Payload .....	82
2211	Table 165: Put Message Payload .....	83
2212	Table 166: Protocol Version Structure in Message Header.....	84
2213	Table 167: Operation in Batch Item .....	84
2214	Table 168: Maximum Response Size in Message Request Header .....	84
2215	Table 169: Unique Batch Item ID in Batch Item.....	85
2216	Table 170: Time Stamp in Message Header .....	85
2217	Table 171: Authentication Structure in Message Header .....	85
2218	Table 172: Asynchronous Indicator in Message Request Header.....	85
2219	Table 173: Asynchronous Correlation Value in Response Batch Item.....	85
2220	Table 174: Result Status in Response Batch Item .....	86
2221	Table 175: Result Reason in Response Batch Item.....	87
2222	Table 176: Result Message in Response Batch Item.....	87
2223	Table 177: Batch Order Option in Message Request Header .....	87
2224	Table 178: Batch Error Continuation Option in Message Request Header .....	87
2225	Table 179: Batch Count in Message Header .....	88
2226	Table 180: Batch Item in Message .....	88
2227	Table 181: Message Extension Structure in Batch Item.....	88
2228	Table 182: Request Message Structure .....	89
2229	Table 183: Response Message Structure.....	89
2230	Table 184: Synchronous Request Header Structure .....	89
2231	Table 185: Synchronous Request Batch Item Structure.....	90
2232	Table 186: Synchronous Response Header Structure .....	90
2233	Table 187: Synchronous Response Batch Item Structure.....	90
2234	Table 188: Asynchronous Request Header Structure .....	91
2235	Table 189: Asynchronous Request Batch Item Structure.....	91
2236	Table 190: Asynchronous Response Header Structure .....	91
2237	Table 191: Asynchronous Response Batch Item Structure.....	92
2238	Table 192: Allowed Item Type Values .....	94
2239	Table 193: Allowed Item Length Values .....	95
2240	Table 194: Tag Values .....	101
2241	Table 195: Credential Type Enumeration .....	102
2242	Table 196: Key Compression Type Enumeration .....	102
2243	Table 197: Key Format Type Enumeration .....	103
2244	Table 198: Wrapping Method Enumeration .....	103
2245	Table 199: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV .....	104
2246	Table 200: Certificate Type Enumeration .....	104
2247	Table 201: Split Key Method Enumeration .....	104
2248	Table 202: Secret Data Type Enumeration.....	105
2249	Table 203: Opaque Data Type Enumeration .....	105
2250	Table 204: Name Type Enumeration .....	105
2251	Table 205: Object Type Enumeration .....	105

2252	Table 206: Cryptographic Algorithm Enumeration .....	106
2253	Table 207: Block Cipher Mode Enumeration .....	107
2254	Table 208: Padding Method Enumeration .....	107
2255	Table 209: Hashing Algorithm Enumeration .....	108
2256	Table 210: Role Type Enumeration .....	109
2257	Table 211: State Enumeration .....	110
2258	Table 212: Revocation Reason Code Enumeration .....	110
2259	Table 213: Link Type Enumeration .....	110
2260	Table 214: Derivation Method Enumeration .....	111
2261	Table 215: Certificate Request Type Enumeration .....	111
2262	Table 216: Validity Indicator Enumeration .....	111
2263	Table 217: Query Function Enumeration .....	112
2264	Table 218: Cancellation Result Enumeration.....	112
2265	Table 219: Put Function Enumeration .....	112
2266	Table 220: Operation Enumeration.....	113
2267	Table 221: Result Status Enumeration .....	114
2268	Table 222: Result Reason Enumeration .....	114
2269	Table 223: Batch Error Continuation Enumeration .....	115
2270	Table 224: Cryptographic Usage Mask.....	115
2271	Table 225: Storage Status Mask.....	116
2272	Table 226: General Errors.....	118
2273	Table 227: Create Errors.....	119
2274	Table 228: Create Key Pair Errors.....	119
2275	Table 229: Register Errors .....	120
2276	Table 230: Re-key Errors .....	120
2277	Table 231: Derive Key Errors-.....	121
2278	Table 232: Certify Errors .....	122
2279	Table 233: Re-certify Errors .....	122
2280	Table 234: Locate Errors.....	122
2281	Table 235: Check Errors .....	122
2282	Table 236: Get Errors.....	123
2283	Table 237: Get Attributes Errors .....	123
2284	Table 238: Get Attribute List Errors .....	124
2285	Table 239: Add Attribute Errors .....	124
2286	Table 240: Modify Attribute Errors .....	125
2287	Table 241: Delete Attribute Errors .....	125
2288	Table 242: Obtain Lease Errors.....	125
2289	Table 243: Get Usage Allocation Errors .....	126
2290	Table 244: Activate Errors.....	126
2291	Table 245: Revoke Errors .....	126
2292	Table 246: Destroy Errors.....	126
2293	Table 247: Archive Errors .....	127

2294	Table 248: Recover Errors .....	127
2295	Table 249: Validate Errors .....	127
2296	Table 250: Poll Errors .....	127
2297	Table 251: Batch Items Errors .....	128
2298	Table 252: Attribute Cross-reference.....	132
2299	Table 253: Tag Cross-reference .....	137
2300	Table 254: Operation and Object Cross-reference.....	138
2301		

2302

---

## F. Acknowledgements

2303 The following individuals have participated in the creation of this specification and are gratefully  
2304 acknowledged:

2305 **Original Authors of the initial contribution:**

2306 David Babcock, HP  
2307 Steven Bade, IBM  
2308 Paolo Bezoari, NetApp  
2309 Mathias Björkqvist, IBM  
2310 Bruce Brinson, EMC  
2311 Christian Cachin, IBM  
2312 Tony Crossman, Thales/nCipher  
2313 Stan Feather, HP  
2314 Indra Fitzgerald, HP  
2315 Judy Furlong, EMC  
2316 Jon Geater, Thales/nCipher  
2317 Bob Griffin, EMC  
2318 Robert Haas, IBM (editor)  
2319 Timothy Hahn, IBM  
2320 Jack Harwood, EMC  
2321 Walt Hubis, LSI  
2322 Glen Jaquette, IBM  
2323 Jeff Kravitz, IBM (editor emeritus)  
2324 Michael McIntosh, IBM  
2325 Brian Metzger, HP  
2326 Anthony Nadalin, IBM  
2327 Elaine Palmer, IBM  
2328 Joe Pato, HP  
2329 René Pawlitzek, IBM  
2330 Subhash Sankuratipati, NetApp  
2331 Mark Schiller, HP  
2332 Martin Skagen, Brocade  
2333 Marcus Streets, Thales/nCipher  
2334 John Tattan, EMC  
2335 Karla Thomas, Brocade  
2336 Marko Vukolić, IBM  
2337 Steve Wierenga, HP

2338 **Participants:**

2339 Gordon Arnold, IBM  
2340 Todd Arnold, IBM  
2341 Matthew Ball, Sun Microsystems  
2342 Elaine Barker, NIST  
2343 Peter Bartok, Venafi, Inc.  
2344 Mathias Björkqvist, IBM  
2345 Kevin Bocek, Thales e-Security  
2346 Kelley Burgin, National Security Agency  
2347 Jon Callas, PGP Corporation  
2348 Tom Clifford, Symantec Corp.  
2349 Graydon Dodson, Lexmark International Inc.  
2350 Chris Dunn, SafeNet, Inc.  
2351 Paul Earsy, SafeNet, Inc.  
2352 Stan Feather, HP  
2353 Indra Fitzgerald, HP  
2354 Alan Frindell, SafeNet, Inc.

2355 Judith Furlong, EMC Corporation  
2356 Jonathan Geater, Thales e-Security  
2357 Robert Griffin, EMC Corporation  
2358 Robert Haas, IBM  
2359 Thomas Hardjono, M.I.T.  
2360 Marc Hocking, BeCrypt Ltd.  
2361 Larry Hofer, Emulex Corporation  
2362 Brandon Hoff, Emulex Corporation  
2363 Walt Hubis, LSI Corporation  
2364 Wyllys Ingersoll, Sun Microsystems  
2365 Jay Jacobs, Target Corporation  
2366 Glen Jaquette, IBM  
2367 Scott Kipp, Brocade Communications Systems, Inc.  
2368 David Lawson, Emulex Corporation  
2369 Robert Lockhart, Thales e-Security  
2370 Shyam Mankala, EMC Corporation  
2371 Marc Massar, Individual  
2372 Don McAlister, Associate  
2373 Hyrum Mills, Mitre Corporation  
2374 Landon Noll, Cisco Systems, Inc.  
2375 René Pawlitzek, IBM  
2376 Rob Philpott, EMC Corporation  
2377 Bruce Rich, IBM  
2378 Scott Rotondo, Sun Microsystems  
2379 Anil Saldhana, Red Hat  
2380 Subhash Sankuratipati, NetApp  
2381 Mark Schiller, HP  
2382 Jitendra Singh, Brocade Communications Systems, Inc.  
2383 Servesh Singh, EMC Corporation  
2384 Sandy Stewart, Sun Microsystems  
2385 Marcus Streets, Thales e-Security  
2386 Brett Thompson, SafeNet, Inc.  
2387 Benjamin Tomhave, Individual  
2388 Sean Turner, IECA, Inc.  
2389 Paul Turner, Venafi, Inc.  
2390 Marko Vukolic, IBM  
2391 Rod Wideman, Quantum Corporation  
2392 Steven Wierenga, HP  
2393 Peter Yee, EMC Corporation  
2394 Krishna Yellepeddy, IBM  
2395 Peter Zelechowski, Associate

## G. Revision History

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

			Furlong) First set of editorial changes as suggested by Elaine Barker (changed Octet to Byte, etc).  (version approved as TC Committee Draft on Sep 29 2009, counts as draft-01 version)
draft-02	2009-10-09	Robert Haas, Indra Fitzgerald	Second set of editorial changes as suggested by Elaine Barker (incl. renaming of "Last Change Date" attribute). Added list of references from Sean Turner and Judy Furlong, as well as terminology. Made Result Reasons in error cases (Sec 11) normative. Added statement on deletion of attributes by server (line 457). Added major/minor 1.0 for protocol version (line 27). Systematic use of <i>italics</i> when introducing a term for first time. Added "Editor's note" comments remaining to be addressed before public review.
draft-03	2009-10-14	Robert Haas, Indra Fitzgerald	Addressed outstanding "Editor's note" comments. Added acronyms and references.
draft-04	2009-10-21	Robert Haas, Indra Fitzgerald	Added the list of participants (Appendix F). Point to the KMIP Profiles document for a list standard application namespaces. Added Terminology (from Bob Lockhart, borrowed from SP800-57 Part 1). Modified title page.
draft-05	2009-11-06	Robert Haas	Additions to the tags table. Added Last Change Date attribute to conformance clause (sec 12.1). Minor edits. This is the tentative revision for public review.
draft-06	2009-11-09	Robert Haas	Editorial fixes to the reference sections. Correction of the comments for the Unique Batch Item ID in the Response Header structures (from Steve Wierenga).

2397