

Key Management Interoperability Protocol Specification Version 1.0

Committee Specification 01

15 Jun 2010

Specification URI

This Version:

<http://docs.oasis-open.org/kmip/spec/v1.0/cs01/kmip-spec-1.0-cs-01.html>
<http://docs.oasis-open.org/kmip/spec/v1.0/cs01/kmip-spec-1.0-cs-01.doc> (Authoritative)
<http://docs.oasis-open.org/kmip/spec/v1.0/cs01/kmip-spec-1.0-cs-01.pdf>

Previous Version:

<http://docs.oasis-open.org/kmip/spec/v1.0/cd12/kmip-spec-1.0-cd-12.html>
<http://docs.oasis-open.org/kmip/spec/v1.0/cd12/kmip-spec-1.0-cd-12.doc> (Authoritative)
<http://docs.oasis-open.org/kmip/spec/v1.0/cd12/kmip-spec-1.0-cd-12.pdf>

Latest Version:

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

Technical Committee:

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

Chair(s):

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

Editor(s):

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

Related work:

This specification replaces or supersedes:

- None

This specification is related to:

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

Declared XML Namespace(s):

None

Abstract:

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

Status:

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

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at <http://www.oasis-open.org/committees/kmip/>.

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

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

Notices

Copyright © OASIS® 2010. 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	14
2	Objects	15
2.1	Base Objects	15
2.1.1	Attribute	15
2.1.2	Credential	16
2.1.3	Key Block.....	16
2.1.4	Key Value	17
2.1.5	Key Wrapping Data	18
2.1.6	Key Wrapping Specification.....	19
2.1.7	Transparent Key Structures.....	20
2.1.8	Template-Attribute Structures.....	25
2.2	Managed Objects.....	25
2.2.1	Certificate	25
2.2.2	Symmetric Key	26
2.2.3	Public Key.....	26
2.2.4	Private Key	26
2.2.5	Split Key	26
2.2.6	Template.....	28
2.2.7	Secret Data.....	29
2.2.8	Opaque Object	29
3	Attributes	30
3.1	Unique Identifier.....	31
3.2	Name	32
3.3	Object Type	32
3.4	Cryptographic Algorithm	33
3.5	Cryptographic Length.....	33
3.6	Cryptographic Parameters	34
3.7	Cryptographic Domain Parameters.....	35
3.8	Certificate Type.....	36
3.9	Certificate Identifier	36
3.10	Certificate Subject.....	37
3.11	Certificate Issuer.....	38
3.12	Digest.....	38
3.13	Operation Policy Name	39
3.13.1	Operations outside of operation policy control	40
3.13.2	Default Operation Policy.....	40
3.14	Cryptographic Usage Mask.....	43
3.15	Lease Time	44
3.16	Usage Limits	45
3.17	State	46

3.18	Initial Date.....	48
3.19	Activation Date.....	48
3.20	Process Start Date.....	49
3.21	Protect Stop Date	50
3.22	Deactivation Date	51
3.23	Destroy Date.....	51
3.24	Compromise Occurrence Date	52
3.25	Compromise Date.....	52
3.26	Revocation Reason.....	53
3.27	Archive Date	53
3.28	Object Group	54
3.29	Link	54
3.30	Application Specific Information.....	56
3.31	Contact Information.....	56
3.32	Last Change Date.....	57
3.33	Custom Attribute	57
4	Client-to-Server Operations.....	59
4.1	Create	59
4.2	Create Key Pair	60
4.3	Register	62
4.4	Re-key	63
4.5	Derive Key	65
4.6	Certify	68
4.7	Re-certify	69
4.8	Locate	71
4.9	Check.....	72
4.10	Get.....	74
4.11	Get Attributes.....	74
4.12	Get Attribute List	75
4.13	Add Attribute	75
4.14	Modify Attribute.....	76
4.15	Delete Attribute	76
4.16	Obtain Lease	77
4.17	Get Usage Allocation	78
4.18	Activate.....	79
4.19	Revoke.....	79
4.20	Destroy	79
4.21	Archive.....	80
4.22	Recover	80
4.23	Validate.....	81
4.24	Query.....	82
4.25	Cancel.....	83
4.26	Poll.....	83
5	Server-to-Client Operations.....	84
5.1	Notify.....	84

5.2	Put	84
6	Message Contents	86
6.1	Protocol Version	86
6.2	Operation	86
6.3	Maximum Response Size	86
6.4	Unique Batch Item ID	86
6.5	Time Stamp	87
6.6	Authentication	87
6.7	Asynchronous Indicator	87
6.8	Asynchronous Correlation Value	87
6.9	Result Status	88
6.10	Result Reason	88
6.11	Result Message	89
6.12	Batch Order Option	89
6.13	Batch Error Continuation Option	89
6.14	Batch Count	90
6.15	Batch Item	90
6.16	Message Extension	90
7	Message Format	91
7.1	Message Structure	91
7.2	Operations	91
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	117
10	Transport	118
11	Error Handling	119
11.1	General	119
11.2	Create	120
11.3	Create Key Pair	120
11.4	Register	121
11.5	Re-key	121
11.6	Derive Key	122
11.7	Certify	123
11.8	Re-certify	123
11.9	Locate	123
11.10	Check	124
11.11	Get	124
11.12	Get Attributes	125
11.13	Get Attribute List	125
11.14	Add Attribute	125
11.15	Modify Attribute	126

11.16	Delete Attribute.....	126
11.17	Obtain Lease	127
11.18	Get Usage Allocation.....	127
11.19	Activate.....	127
11.20	Revoke	128
11.21	Destroy	128
11.22	Archive	128
11.23	Recover	128
11.24	Validate	128
11.25	Query.....	129
11.26	Cancel	129
11.27	Poll	129
11.28	Batch Items.....	129
12	Server Baseline Implementation Conformance Profile	130
12.1	Conformance clauses for a KMIP Server	130
A.	Attribute Cross-reference.....	132
B.	Tag Cross-reference.....	134
C.	Operation and Object Cross-reference.....	139
D.	Acronyms	140
E.	List of Figures and Tables	143
F.	Acknowledgements.....	150
G.	Revision History	152

1 Introduction

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

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

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

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

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

Archive	To place information not accessed frequently into long-term storage.
Asymmetric key pair (key pair)	A public key and its corresponding private key; a key pair is used with a public key algorithm.
Authentication	A process that establishes the origin of information, or determines an entity's identity.
Authentication code	A cryptographic checksum based on a security function (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"> 1. The transformation of plaintext data into ciphertext data, 2. The transformation of ciphertext data into plaintext data, 3. The computation of a digital signature from data, 4. The verification of a digital signature, 5. The computation of an authentication code from data, 6. The verification of an authentication code from data and a received authentication code.
Decryption	The process of changing ciphertext into plaintext using a cryptographic algorithm and key.
Digest (or hash)	The result of applying a hashing algorithm to information.
Digital signature (signature)	The result of a cryptographic transformation of data that, when properly implemented with supporting infrastructure and policy, provides the services of: <ol style="list-style-type: none"> 1. origin authentication 2. data integrity, and 3. signer non-repudiation.
Encryption	The process of changing plaintext into ciphertext using a cryptographic algorithm and key.
Hashing algorithm (or hash algorithm, hash function)	An algorithm that maps a bit string of arbitrary length to a fixed length bit string. Approved hashing algorithms satisfy the following properties: <ol style="list-style-type: none"> 1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and 2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output.
Integrity	The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner.
Key derivation (derivation)	A function in the lifecycle of keying material; the process by which one or more keys are derived from 1) either a shared secret from a key agreement computation or a pre-shared cryptographic key, and 2) other

	information.
Key management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., IVs and passwords) during the entire life cycle of the keys, including their generation, storage, establishment, entry and output, and destruction.
Key wrapping (wrapping)	A method of encrypting and/or MACing/signing keys.
Message authentication code (MAC)	A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data.
PGP certificate	A transferable public key in the OpenPGP Message Format (see [RFC4880]).
Private key	A cryptographic key, used with a public key cryptographic algorithm, that is uniquely associated with an entity and is not made public. The private key is associated with a public key. Depending on the algorithm, the private key may be used to: <ol style="list-style-type: none"> 1. Compute the corresponding public key, 2. Compute a digital signature that may be verified by the corresponding public key, 3. Decrypt data that was encrypted by the corresponding public key, or 4. Compute a piece of common shared data, together with other information.
Profile	A specification of objects, attributes, operations, message elements and authentication methods to be used in specific contexts of key management server and client interactions (see [KMIP-Prof]).
Public key	A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that may be made public. 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"> 1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt data that can be decrypted by the corresponding private key, or 3. Compute a piece of shared data.
Public key certificate (certificate)	A set of data that uniquely identifies an entity, contains the entity's public key and possibly other information, and is digitally signed by a trusted party, thereby binding the public key to the entity.
Public key cryptographic algorithm	A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible.
Public Key Infrastructure	A framework that is established to issue, maintain and revoke public key certificates.
Recover	To retrieve information that was archived to long-term storage.
Split knowledge	A process by which a cryptographic key is split into n multiple key components, individually providing no knowledge of the original key, which can be subsequently combined to recreate the original

	cryptographic key. If knowledge of k (where k is less than or equal to n) components is required to construct the original key, then knowledge of any $k-1$ key components provides no information about the original key 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-forged 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.

35

Table 1: Terminology

36 1.2 Normative References

- 37 [FIPS186-3] *Digital Signature Standard (DSS)*, FIPS PUB 186-3, Jun 2009,
38 http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf
- 39 [FIPS197] *Advanced Encryption Standard*, FIPS PUB 197, Nov 2001,
40 <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- 41 [FIPS198-1] *The Keyed-Hash Message Authentication Code (HMAC)*, FIPS PUB 198-1, Jul
42 2008, http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
- 43 [IEEE1003-1] IEEE Std 1003.1, *Standard for information technology - portable operating
44 system interface (POSIX). Shell and utilities*, 2004.
- 45 [ISO16609] ISO, *Banking -- Requirements for message authentication using symmetric
46 techniques*, ISO 16609, 1991
- 47 [ISO9797-1] ISO/IEC, *Information technology -- Security techniques -- Message
48 Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher*,
49 ISO/IEC 9797-1, 1999
- 50 [KMIP-Prof] OASIS Committee Specification 01, *Key Management Interoperability Protocol
51 Profiles Version 1.0*, June 2010, [http://docs.oasis-
52 open.org/kmip/profiles/v1.0/cs01/kmip-profiles-1.0-cs-01.doc](http://docs.oasis-open.org/kmip/profiles/v1.0/cs01/kmip-profiles-1.0-cs-01.doc)
- 53 [PKCS#1] RSA Laboratories, *PKCS #1 v2.1: RSA Cryptography Standard*, Jun 14, 2002,
54 <http://www.rsa.com/rsalabs/node.asp?id=2125>
- 55 [PKCS#5] RSA Laboratories, *PKCS #5 v2.1: Password-Based Cryptography Standard*, Oct
56 5, 2006, <http://www.rsa.com/rsalabs/node.asp?id=2127>
- 57 [PKCS#7] RSA Laboratories, *PKCS#7 v1.5: Cryptographic Message Syntax Standard*, Nov
58 1, 1993, <http://www.rsa.com/rsalabs/node.asp?id=2129>
- 59 [PKCS#8] RSA Laboratories, *PKCS#8 v1.2: Private-Key Information Syntax Standard*, Nov
60 1, 1993, <http://www.rsa.com/rsalabs/node.asp?id=2130>
- 61 [PKCS#10] RSA Laboratories, *PKCS #10 v1.7: Certification Request Syntax Standard*, May
62 26, 2000, <http://www.rsa.com/rsalabs/node.asp?id=2132>
- 63 [RFC1319] B. Kaliski, *The MD2 Message-Digest Algorithm*, IETF RFC 1319, Apr 1992,
64 <http://www.ietf.org/rfc/rfc1319.txt>
- 65 [RFC1320] R. Rivest, *The MD4 Message-Digest Algorithm*, IETF RFC 1320, Apr 1992,
66 <http://www.ietf.org/rfc/rfc1320.txt>

- 67 [RFC1321] R. Rivest, *The MD5 Message-Digest Algorithm*, IETF RFC 1321, Apr 1992,
68 <http://www.ietf.org/rfc/rfc1321.txt>
- 69 [RFC1421] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I: Message*
70 *Encryption and Authentication Procedures*, IETF RFC 1421, Feb 1993,
71 <http://www.ietf.org/rfc/rfc1421.txt>
- 72 [RFC1424] B. Kaliski, *Privacy Enhancement for Internet Electronic Mail: Part IV: Key*
73 *Certification and Related Services*, IETF RFC 1424, Feb 1993,
74 <http://www.ietf.org/rfc/rfc1424.txt>
- 75 [RFC2104] H. Krawczyk, M. Bellare, R. Canetti, *HMAC: Keyed-Hashing for Message*
76 *Authentication*, IETF RFC 2104, Feb 1997, <http://www.ietf.org/rfc/rfc2104.txt>
- 77 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF
78 RFC 2119, Mar 1997, <http://www.ietf.org/rfc/rfc2119.txt>
- 79 [RFC 2246] T. Dierks and C. Allen, *The TLS Protocol, Version 1.0*, IETF RFC 2246, Jan
80 1999, <http://www.ietf.org/rfc/rfc2246.txt>
- 81 [RFC2898] B. Kaliski, *PKCS #5: Password-Based Cryptography Specification Version 2.0*,
82 IETF RFC 2898, Sep 2000, <http://www.ietf.org/rfc/rfc2898.txt>
- 83 [RFC 3394] J. Schaad, R. Housley, *Advanced Encryption Standard (AES) Key Wrap*
84 *Algorithm*, IETF RFC 3394, Sep 2002, <http://www.ietf.org/rfc/rfc3394.txt>
- 85 [RFC3447] J. Jonsson, B. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA*
86 *Cryptography Specifications Version 2.1*, IETF RFC 3447, Feb 2003,
87 <http://www.ietf.org/rfc/rfc3447.txt>
- 88 [RFC3629] F. Yergeau, *UTF-8, a transformation format of ISO 10646*, IETF RFC 3629, Nov
89 2003, <http://www.ietf.org/rfc/rfc3629.txt>
- 90 [RFC3647] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, *Internet X.509 Public Key*
91 *Infrastructure Certificate Policy and Certification Practices Framework*, IETF RFC
92 3647, Nov 2003, <http://www.ietf.org/rfc/rfc3647.txt>
- 93 [RFC4210] C. Adams, S. Farrell, T. Kause and T. Mononen, *Internet X.509 Public Key*
94 *Infrastructure Certificate Management Protocol (CMP)*, IETF RFC 2510, Sep
95 2005, <http://www.ietf.org/rfc/rfc4210.txt>
- 96 [RFC4211] J. Schaad, *Internet X.509 Public Key Infrastructure Certificate Request Message*
97 *Format (CRMF)*, IETF RFC 4211, Sep 2005, <http://www.ietf.org/rfc/rfc4211.txt>
- 98 [RFC4868] S. Kelly, S. Frankel, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-*
99 *512 with IPsec*, IETF RFC 4868, May 2007, <http://www.ietf.org/rfc/rfc4868.txt>
- 100 [RFC4880] J. Callas, L. Donnerhackle, H. Finney, D. Shaw, and R. Thayer, *OpenPGP*
101 *Message Format*, IETF RFC 4880, Nov 2007, <http://www.ietf.org/rfc/rfc4880.txt>
- 102 [RFC4949] R. Shirey, *Internet Security Glossary, Version 2*, IETF RFC 4949, Aug 2007,
103 <http://www.ietf.org/rfc/rfc4949.txt>
- 104 [RFC5272] J. Schaad and M. Meyers, *Certificate Management over CMS (CMC)*, IETF RFC
105 5272, Jun 2008, <http://www.ietf.org/rfc/rfc5272.txt>
- 106 [RFC5280] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, *Internet*
107 *X.509 Public Key Infrastructure Certificate*, IETF RFC 5280, May 2008,
108 <http://www.ietf.org/rfc/rfc5280.txt>
- 109 [RFC5649] R. Housley, *Advanced Encryption Standard (AES) Key Wrap with Padding*
110 *Algorithm*, IETF RFC 5649, Aug 2009, <http://www.ietf.org/rfc/rfc5649.txt>
- 111 [SHAMIR1979] A. Shamir, *How to share a secret*, Communications of the ACM, vol. 22, no. 11,
112 pp. 612-613, Nov 1979
- 113 [SP800-38A] M. Dworkin, *Recommendation for Block Cipher Modes of Operation – Methods*
114 *and Techniques*, NIST Special Publication 800-38A, Dec 2001,
115 <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>
- 116 [SP800-38B] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CMAC*
117 *Mode for Authentication*, NIST Special Publication 800-38B, May 2005,
118 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

119 [SP800-38C] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: the CCM*
120 *Mode for Authentication and Confidentiality*, NIST Special Publication 800-38C,
121 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)
122 [38C_updated-July20_2007.pdf](http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf)

123 [SP800-38D] M. Dworkin, *Recommendation for Block Cipher Modes of Operation:*
124 *Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication 800-38D, Nov
125 2007, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>

126 [SP800-38E] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The XTS-*
127 *AES Mode for Confidentiality on Block-Oriented Storage Devices*, NIST Special
128 Publication 800-38E, Jan 2010, [http://csrc.nist.gov/publications/nistpubs/800-](http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf)
129 [38E/nist-sp-800-38E.pdf](http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf)

130 [SP800-56A] E. Barker, D. Johnson, and M. Smid, *Recommendation for Pair-Wise Key*
131 *Establishment Schemes Using Discrete Logarithm Cryptography (Revised)*, NIST
132 Special Publication 800-56A, Mar 2007,
133 [http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-](http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf)
134 [2007.pdf](http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf)

135 [SP800-56B] E. Barker, L. Chen, A. Regenscheid, and M. Smid, *Recommendation for Pair-*
136 *Wise Key Establishment Schemes Using Integer Factorization Cryptography*,
137 NIST Special Publication 800-56B, Aug 2009,
138 <http://csrc.nist.gov/publications/nistpubs/800-56B/sp800-56B.pdf>

139 [SP800-57-1] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendations for Key*
140 *Management - Part 1: General (Revised)*, NIST Special Publication 800-57 part
141 1, Mar 2007, [http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-](http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf)
142 [revised2_Mar08-2007.pdf](http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf)

143 [SP800-67] W. Barker, *Recommendation for the Triple Data Encryption Algorithm (TDEA)*
144 *Block Cipher*, NIST Special Publication 800-67, Version 1.1, Revised 19 May
145 2008, <http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf>

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

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

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

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

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

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

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

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

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

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

169

170 **1.3 Non-normative References**

171 **[KMIP-UG]** OASIS Committee Specification 01, Key Management Interoperability Protocol
172 Usage Guide Version 1.0, June 2010, [http://docs.oasis-
open.org/kmip/ug/v1.0/cs01/kmip-ug-1.0-cs-01.doc](http://docs.oasis-
173 open.org/kmip/ug/v1.0/cs01/kmip-ug-1.0-cs-01.doc)

174 **[KMIP-UC]** Committee Specification 01, Key Management Interoperability Protocol Use
175 Cases Version 1.0, June 2010, [http://docs.oasis-
open.org/kmip/usecases/v1.0/cs01/kmip-usecases-1.0-cs-01.doc](http://docs.oasis-
176 open.org/kmip/usecases/v1.0/cs01/kmip-usecases-1.0-cs-01.doc)

177 **[ISO/IEC 9945-2]** The Open Group, *Regular Expressions, The Single UNIX Specification version 2*,
178 1997, ISO/IEC 9945-2:1993,
179 <http://www.opengroup.org/onlinepubs/007908799/xbd/re.html>

180

181 2 Objects

182 The following subsections describe the objects that are passed between the clients and servers of the key
183 management system. Some of these object types, called *Base Objects*, are used only in the protocol
184 itself, and are not considered Managed Objects. Key management systems MAY choose to support a
185 subset of the Managed Objects. The object descriptions refer to the primitive data types of which they are
186 composed. These primitive data types are (see Section 9.1.1.4):

- 187 • Integer
- 188 • Long Integer
- 189 • Big Integer
- 190 • Enumeration – choices from a predefined list of values
- 191 • Boolean
- 192 • Text String – string of characters representing human-readable text
- 193 • Byte String – sequence of unencoded byte values
- 194 • Date-Time – date and time, with a granularity of one second
- 195 • Interval – a length of time expressed in seconds

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

197 2.1 Base Objects

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

200 2.1.1 Attribute

201 An Attribute object is a structure (see Table 2) used for sending and receiving Managed Object attributes.
202 The *Attribute Name* is a text-string that is used to identify the attribute. The *Attribute Index* is an index
203 number assigned by the key management server when a specified named attribute is allowed to have
204 multiple instances. The Attribute Index is used to identify the particular instance. Attribute Indices SHALL
205 start with 0. The Attribute Index of an attribute SHALL NOT change when other instances are added or
206 deleted. For example, if a particular attribute has 4 instances with Attribute Indices 0, 1, 2 and 3, and the
207 instance with Attribute Index 2 is deleted, then the Attribute Index of instance 3 is not changed. Attributes
208 that have a single instance have an Attribute Index of 0, which is assumed if the Attribute Index is not
209 specified. The *Attribute Value* is either a primitive data type or structured object, depending on the
210 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, except for the Notify operation (see Section 5.1)

211 **Table 2: Attribute Object Structure**

212 **2.1.2 Credential**

213 A *Credential* is a structure (see Table 3) used for client identification purposes and is not managed by the
 214 key management system (e.g., user id/password pairs, Kerberos tokens, etc). It MAY be used for
 215 authentication purposes as indicated in [KMIP-Prof].

Object	Encoding	REQUIRED
Credential	Structure	
Credential Type	Enumeration, see 9.1.3.2.1	Yes
Credential Value	Varies. Structure for Username and Password Credential Type.	Yes

216 **Table 3: Credential Object Structure**

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

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

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

221
 222 **2.1.3 Key Block**

223 A *Key Block* object is a structure (see Table 5) used to encapsulate all of the information that is closely
 224 associated with a cryptographic key. It contains a Key Value of one of the following *Key Format Types*:

- 225 • *Raw* – This is a key that contains only cryptographic key material, encoded as a string of bytes.
- 226 • *Opaque* – This is an encoded key for which the encoding is unknown to the key management
 227 system. It is encoded as a string of bytes.
- 228 • *PKCS1* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#1 object.
- 229 • *PKCS8* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#8 object,
 230 supporting both the RSAPrivateKey syntax and EncryptedPrivateKey.
- 231 • *X.509* – This is an encoded object, expressed as a DER-encoded ASN.1 X.509 object.
- 232 • *ECPrivateKey* – This is an ASN.1 encoded elliptic curve private key.
- 233 • *Several Transparent Key types* – These are algorithm-specific structures containing defined
 234 values for the various key types, as defined in Section 2.1.7
- 235 • *Extensions* – These are vendor-specific extensions to allow for proprietary or legacy key formats.

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

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

- 240 • RSA keys are typically 1024, 2048 or 3072 bits in length

- 241 • 3DES keys are typically from 112 to 192 bits (depending upon key length and the presence of
- 242 parity bits)
- 243 • AES keys are 128, 192 or 256 bits in length

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

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

246 **Table 5: Key Block Object Structure**

247 2.1.4 Key Value

248 The *Key Value* is used only inside a Key Block and is either a Byte String or a structure (see Table 6):

- 249 • The Key Value structure contains the key material, either as a byte string or as a Transparent Key
- 250 structure (see Section 2.1.7), and OPTIONAL attribute information that is associated and
- 251 encapsulated with the key material. This attribute information differs from the attributes
- 252 associated with Managed Objects, and which is obtained via the Get Attributes operation, only by
- 253 the fact that it is encapsulated with (and possibly wrapped with) the key material itself.
- 254 • 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

255 **Table 6: Key Value Object Structure**

256 **2.1.5 Key Wrapping Data**

257 The Key Block MAY also supply OPTIONAL information about a cryptographic key wrapping mechanism
 258 used to wrap the Key Value. This consists of a *Key Wrapping Data* structure (see Table 7). It is only used
 259 inside a Key Block.

260 This structure contains fields for:

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

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

275 The following wrapping methods are currently defined:

- 276 • *Encrypt* only (i.e., encryption using a symmetric key or public key, or authenticated encryption
 277 algorithms that use a single key)
- 278 • *MAC/sign* only (i.e., either MACing the Key Value with a symmetric key, or signing the Key Value
 279 with a private key)
- 280 • *Encrypt then MAC/sign*
- 281 • *MAC/sign then encrypt*
- 282 • *TR-31*
- 283 • *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

284 **Table 7: Key Wrapping Data Object Structure**

285 The structures of the Encryption Key Information (see Table 8) and the MAC/Signature Key Information
 286 (see Table 9) 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

287 **Table 8: 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

288 **Table 9: MAC/Signature Key Information Object Structure**

289 2.1.6 Key Wrapping Specification

290 This is a separate structure (see Table 10) that is defined for operations that provide the option to return
 291 wrapped keys. The *Key Wrapping Specification* SHALL be included inside the operation request if clients
 292 request the server to return a wrapped key. If Cryptographic Parameters are specified in the Encryption
 293 Key Information and/or the MAC/Signature Key Information of the Key Wrapping Specification, then the
 294 server SHALL verify that they match one of the instances of the Cryptographic Parameters attribute of the
 295 corresponding key. If Cryptographic Parameters are omitted, then the server SHALL use the
 296 Cryptographic Parameters attribute with the lowest Attribute Index of the corresponding key. If the

297 corresponding key does not have any Cryptographic Parameters attribute, or if no match is found, then an
 298 error is returned.

299 This structure contains:

- 300 • A Wrapping Method that indicates the method used to wrap the Key Value.
- 301 • Encryption Key Information with the Unique Identifier value of the encryption key and associated
 302 cryptographic parameters.
- 303 • MAC/Signature Key Information with the Unique Identifier value of the MAC/signature key and
 304 associated cryptographic parameters.
- 305 • 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

306 **Table 10: Key Wrapping Specification Object Structure**

307 2.1.7 Transparent Key Structures

308 *Transparent Key* structures describe the necessary parameters to obtain the key material. They are used
 309 in the Key Value structure. The mapping to the parameters specified in other standards is shown in Table
 310 11.

Object	Description	Mapping
P	For DSA and DH, the (large) prime field order. For RSA, a prime factor of the modulus.	p in [FIPS186-3], [X9.42], [SP800-56A] p in [PKCS#1], [SP800-56B]
Q	For DSA and DH, the (small) prime multiplicative subgroup order. For RSA, a prime factor of the modulus.	q in [FIPS186-3], [X9.42], [SP800-56A] q in [PKCS#1], [SP800-56B]
G	The generator of the subgroup of order Q.	g in [FIPS186-3], [X9.42], [SP800-56A]
X	DSA or DH private key.	x in [FIPS186-3] x, x _u , x _v in [X9.42], [SP800-56A] for static private keys r, r _u , r _v in [X9.42], [SP800-56A] for ephemeral private keys
Y	DSA or DH public key.	y in [FIPS186-3] y, y _u , y _v in [X9.42], [SP800-

		56A] for static public keys t, t _u , t _v in [X9.42] , [SP800-56A] for ephemeral public keys
J	DH cofactor integer, where P = JQ + 1.	j in [X9.42]
Modulus	RSA modulus PQ, where P and Q are distinct primes.	n in [PKCS#1] , [SP800-56B]
Private Exponent	RSA private exponent.	d in [PKCS#1] , [SP800-56B]
Public Exponent	RSA public exponent.	e in [PKCS#1] , [SP800-56B]
Prime Exponent P	RSA private exponent for the prime factor P in the CRT format, i.e., Private Exponent (mod (P-1)).	dP in [PKCS#1] , [SP800-56B]
Prime Exponent Q	RSA private exponent for the prime factor Q in the CRT format, i.e., Private Exponent (mod (Q-1)).	dQ in [PKCS#1] , [SP800-56B]
CRT Coefficient	The (first) CRT coefficient, i.e., Q ⁻¹ mod P.	qInv in [PKCS#1] , [SP800-56B]
Recommended Curve	NIST Recommended Curves (e.g., P-192).	See Appendix D of [FIPS186-3]
D	Elliptic curve private key.	d; d _{e,U} , d _{e,V} (ephemeral private keys); d _{s,U} , d _{s,V} (static private keys) in [X9-63] , [SP800-56A]
Q String	Elliptic curve public key.	Q; Q _{e,U} , Q _{e,V} (ephemeral public keys); Q _{s,U} , Q _{s,V} (static public keys) in [X9-63] , [SP800-56A]

311 **Table 11: Parameter mapping.**

312 **2.1.7.1 Transparent Symmetric Key**

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

Object	Encoding	REQUIRED
Key Material	Structure	
Key	Byte String	Yes

315 **Table 12: Key Material Object Structure for Transparent Symmetric Keys**

316 **2.1.7.2 Transparent DSA Private Key**

317 If the Key Format Type in the Key Block is *Transparent DSA Private Key*, then Key Material is a structure
318 as shown in Table 13.

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

319 **Table 13: Key Material Object Structure for Transparent DSA Private Keys**

320 2.1.7.3 Transparent DSA Public Key

321 If the Key Format Type in the Key Block is *Transparent DSA Public Key*, then Key Material is a structure
322 as shown in Table 14.

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

323 **Table 14: Key Material Object Structure for Transparent DSA Public Keys**

324 2.1.7.4 Transparent RSA Private Key

325 If the Key Format Type in the Key Block is *Transparent RSA Private Key*, then Key Material is a structure
326 as shown in Table 15.

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

327 **Table 15: Key Material Object Structure for Transparent RSA Private Keys**

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

- 329 • Private Exponent
- 330 • P and Q (the first two prime factors of Modulus)
- 331 • Prime Exponent P and Prime Exponent Q.

332 **2.1.7.5 Transparent RSA Public Key**

333 If the Key Format Type in the Key Block is *Transparent RSA Public Key*, then Key Material is a structure
334 as shown in Table 16.

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

335 **Table 16: Key Material Object Structure for Transparent RSA Public Keys**

336 **2.1.7.6 Transparent DH Private Key**

337 If the Key Format Type in the Key Block is *Transparent DH Private Key*, then Key Material is a structure
338 as shown in Table 17.

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

339 **Table 17: Key Material Object Structure for Transparent DH Private Keys**

340 **2.1.7.7 Transparent DH Public Key**

341 If the Key Format Type in the Key Block is *Transparent DH Public Key*, then Key Material is a structure as
342 shown in Table 18.

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

343 **Table 18: Key Material Object Structure for Transparent DH Public Keys**

344 **2.1.7.8 Transparent ECDSA Private Key**

345 If the Key Format Type in the Key Block is *Transparent ECDSA Private Key*, then Key Material is a
346 structure as shown in Table 19.

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

347 **Table 19: Key Material Object Structure for Transparent ECDSA Private Keys**

348 **2.1.7.9 Transparent ECDSA Public Key**

349 If the Key Format Type in the Key Block is *Transparent ECDSA Public Key*, then Key Material is a
350 structure as shown in Table 20.

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

351 **Table 20: Key Material Object Structure for Transparent ECDSA Public Keys**

352 **2.1.7.10 Transparent ECDH Private Key**

353 If the Key Format Type in the Key Block is *Transparent ECDH Private Key*, then Key Material is a
354 structure as shown in Table 21.

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

355 **Table 21: Key Material Object Structure for Transparent ECDH Private Keys**

356 **2.1.7.11 Transparent ECDH Public Key**

357 If the Key Format Type in the Key Block is *Transparent ECDH Public Key*, then Key Material is a structure
358 as shown in Table 22.

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

359 **Table 22: Key Material Object Structure for Transparent ECDH Public Keys**

360 **2.1.7.12 Transparent ECMQV Private Key**

361 If the Key Format Type in the Key Block is *Transparent ECMQV Private Key*, then Key Material is a
362 structure as shown in Table 23.

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

363 **Table 23: Key Material Object Structure for Transparent ECMQV Private Keys**

364 2.1.7.13 Transparent ECMQV Public Key

365 If the Key Format Type in the Key Block is *Transparent ECMQV Public Key*, then Key Material is a
366 structure as shown in Table 24.

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

367 **Table 24: Key Material Object Structure for Transparent ECMQV Public Keys**

368 2.1.8 Template-Attribute Structures

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

371 The *Template-Attribute*, *Common Template-Attribute*, *Private Key Template-Attribute*, and *Public Key*
372 *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

373 **Table 25: Template-Attribute Object Structure**

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

375 2.2 Managed Objects

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

379 2.2.1 Certificate

380 A Managed Cryptographic Object that is a digital certificate. For X.509 certificates, its is a DER-encoded
381 X.509 public key certificate, For PGP certificates, it is a transferable public key in the OpenPGP message
382 format..

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

383 **Table 26: Certificate Object Structure**

384 2.2.2 Symmetric Key

385 A Managed Cryptographic Object that is a symmetric key.

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

386 **Table 27: Symmetric Key Object Structure**

387 2.2.3 Public Key

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

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

390 **Table 28: Public Key Object Structure**

391 2.2.4 Private Key

392 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

393 **Table 29: Private Key Object Structure**

394 2.2.5 Split Key

395 A Managed Cryptographic Object that is a *Split Key*. A split key is a secret, usually a symmetric key or a
396 private key that has been split into a number of parts, each of which MAY then be distributed to several
397 key holders, for additional security. The *Split Key Parts* field indicates the total number of parts, and the
398 *Split Key Threshold* field indicates the minimum number of parts needed to reconstruct the entire key.
399 The *Key Part Identifier* indicates which key part is contained in the cryptographic object, and SHALL be at
400 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 30: Split Key Object Structure

401

402 There are three *Split Key Methods* for secret sharing: the first one is based on XOR, and the other two
403 are based on polynomial secret sharing, according to [SHAMIR1979].

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

- 405 • When the Split Key Method is XOR, then the Key Material in the Key Value of the Key Block is of
406 length L bits. The number of split keys is Split Key Parts (identical to Split Key Threshold), and
407 the secret is reconstructed by XORing all of the parts.
- 408 • When the Split Key Method is Polynomial Sharing Prime Field, then secret sharing is performed
409 in the field $GF(\text{Prime Field Size})$, represented as integers, where Prime Field Size is a prime
410 bigger than 2^L .
- 411 • When the Split Key Method is Polynomial Sharing $GF(2^{16})$, then secret sharing is performed in
412 the field $GF(2^{16})$. The Key Material in the Key Value of the Key Block is a bit string of length L ,
413 and when L is bigger than 2^{16} , then secret sharing is applied piecewise in pieces of 16 bits each.
414 The Key Material in the Key Value of the Key Block is the concatenation of the corresponding
415 shares of all pieces of the secret.

416 Secret sharing is performed in the field $GF(2^{16})$, which is represented as an algebraic extension of
417 $GF(2^8)$:

418
$$GF(2^{16}) \approx GF(2^8) [y]/(y^2+y+m), \quad \text{where } m \text{ is defined later.}$$

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

421 The representation of field elements and the notation in this section rely on [FIPS197], Sections 3
422 and 4. The field $GF(2^8)$ is as described in [FIPS197],

423
$$GF(2^8) \approx GF(2) [x]/(x^8+x^4+x^3+x+1).$$

424 An element of $GF(2^8)$ is represented as a byte. Addition and subtraction in $GF(2^8)$ is performed as
425 a bit-wise XOR of the bytes. Multiplication and inversion are more complex (see [FIPS197]
426 Section 4.1 and 4.2 for details).

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

428
$$m = x^5+x^4+x^3+x,$$

429 which is represented by the byte 0x3A (or {3A} in notation according to [FIPS197]).

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

432
$$(ry + s)(uy + v) = ((r + s)(u + v) + sv)y + (ru + svm).$$

433 The inverse of an element $uy + v$ is given by
 434 $(uy + v)^{-1} = ud^{-1}y + (u + v)d^{-1}$, where $d = (u + v)v + mu^2$.

435 2.2.6 Template

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

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

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

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

448 Attributes applicable to objects created using the Template are:

- 449 • Cryptographic Algorithm
- 450 • Cryptographic Length
- 451 • Cryptographic Domain Parameters
- 452 • Cryptographic Parameters
- 453 • Operation Policy Name
- 454 • Cryptographic Usage Mask
- 455 • Usage Limits
- 456 • Activation Date
- 457 • Process Start Date
- 458 • Protect Stop Date
- 459 • Deactivation Date
- 460 • Object Group
- 461 • Application Specific Information
- 462 • Contact Information
- 463 • Custom Attribute

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

464 **Table 31: Template Object Structure**

465 **2.2.7 Secret Data**

466 A Managed Cryptographic Object containing a shared secret value that is not a key or certificate (e.g., a
467 password). The Key Block of the *Secret Data* object contains a Key Value of the Opaque type. The Key
468 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

469 **Table 32: Secret Data Object Structure**

470 **2.2.8 Opaque Object**

471 A Managed Object that the key management server is possibly not able to interpret. The context
472 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

473 **Table 33: Opaque Object Structure**

474

3 Attributes

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

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

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

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

493 The second table in each subsection lists certain attribute characteristics (e.g., "SHALL always have a
494 value"): Table 34 below explains the meaning of each characteristic that may appear in those tables. The
495 server policy MAY further restrict these attribute characteristics.

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

496

Table 34: Attribute Rules

497 **3.1 Unique Identifier**

498 The *Unique Identifier* is generated by the key management system to uniquely identify a Managed Object.
 499 It is only REQUIRED to be unique within the identifier space managed by a single key management
 500 system, however it is RECOMMENDED that this identifier be globally unique in order to allow for a key
 501 management domain export of such objects. This attribute SHALL be assigned by the key management
 502 system at creation or registration time, and then SHALL NOT be changed or deleted before the object is
 503 destroyed.

Object	Encoding	
Unique Identifier	Text String	

504

Table 35: 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

505

Table 36: Unique Identifier Attribute Rules

506 3.2 Name

507 The *Name* attribute is a structure (see Table 37) used to identify and locate the object. This attribute is
508 assigned by the client, and the *Name Value* is intended to be in a form that humans are able to interpret.
509 The key management system MAY specify rules by which the client creates valid names. Clients are
510 informed of such rules by a mechanism that is not specified by this standard. Names SHALL be unique
511 within a given key management domain, but are not REQUIRED to be globally unique.

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

512

Table 37: 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

513

Table 38: Name Attribute Rules

514 3.3 Object Type

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

Object	Encoding	
Object Type	Enumeration, see 9.1.3.2.11	

518

Table 39: 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

519

Table 40: Object Type Attribute Rules

520 3.4 Cryptographic Algorithm

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

Object	Encoding	
Cryptographic Algorithm	Enumeration, see 9.1.3.2.12	

524

Table 41: 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

525

Table 42: Cryptographic Algorithm Attribute Rules

526 3.5 Cryptographic Length

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

Object	Encoding	
Cryptographic Length	Integer	

530

Table 43: 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

531 **Table 44: Cryptographic Length Attribute Rules**

532 3.6 Cryptographic Parameters

533 The *Cryptographic Parameters* attribute is a structure (see Table 45) that contains a set of OPTIONAL
534 fields that describe certain cryptographic parameters to be used when performing cryptographic
535 operations using the object. Specific fields MAY pertain only to certain types of Managed Cryptographic
536 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
Key Role Type	Enumeration, see 9.1.3.2.16	No

537 **Table 45: 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

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

539 Key Role Type definitions match those defined in ANSI X9 TR-31 [X9 TR-31] and are defined in Table
540 47:

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

Table 47: Key Role Types

541
542 Accredited Standards Committee X9, Inc. - Financial Industry Standards (www.x9.org) contributed to
543 Table 47. Key role names and descriptions are derived from material in the Accredited Standards
544 Committee X9, Inc's Technical Report "TR-31 2005 Interoperable Secure Key Exchange Key Block
545 Specification for Symmetric Algorithms" and used with the permission of Accredited Standards Committee
546 X9, Inc. in an effort to improve interoperability between X9 standards and OASIS KMIP. The complete
547 ANSI X9 TR-31 is available at www.x9.org.

548 **3.7 Cryptographic Domain Parameters**

549 The *Cryptographic Domain Parameters* attribute is a structure (see Table 48) that contains a set of
550 OPTIONAL fields that MAY need to be specified in the Create Key Pair Request Payload. Specific fields
551 MAY only pertain to certain types of Managed Cryptographic Objects.

552 The domain parameter Qlength corresponds to the bit length of parameter Q (refer to **[FIPS186-3]** and
553 **[SP800-56A]**). Qlength applies to algorithms such as DSA and DH. The bit length of parameter P (refer to
554 **[FIPS186-3]** and **[SP800-56A]**) is specified separately by setting the Cryptographic Length attribute.

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

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

556

Table 48: 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

557

Table 49: Cryptographic Domain Parameters Attribute Rules

558 3.8 Certificate Type

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

Object	Encoding	
Certificate Type	Enumeration, see 9.1.3.2.6	

562

Table 50: 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

563

Table 51: Certificate Type Attribute Rules

564 3.9 Certificate Identifier

565 The *Certificate Identifier* attribute is a structure (see Table 52) used to provide the identification of a
 566 certificate. For X.509 certificates, it contains the Issuer Distinguished Name (i.e., from the Issuer field of
 567 the certificate) and the Certificate Serial Number (i.e., from the Serial Number field of the certificate). For
 568 PGP certificates, the Issuer contains the OpenPGP Key ID of the key issuing the signature (the signature

569 that represents the certificate). The Certificate Identifier SHALL be set by the server when the certificate is
 570 created or registered and then SHALL NOT be changed or deleted before the object is destroyed.

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

571 **Table 52: 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

572 **Table 53: Certificate Identifier Attribute Rules**

573 3.10 Certificate Subject

574 The *Certificate Subject* attribute is a structure (see Table 54) used to identify the subject of a certificate.
 575 For X.509 certificates, it contains the Subject Distinguished Name (i.e., from the Subject field of the
 576 certificate). It MAY include one or more alternative names (e.g., email address, IP address, DNS name)
 577 for the subject of the certificate (i.e., from the Subject Alternative Name extension within the certificate).
 578 For PGP certificates, the Certificate Subject Distinguished Name contains the content of the first User ID
 579 packet in the PGP certificate (that is, the first User ID packet after the Public-Key packet in the
 580 transferable public key that forms the PGP certificate). These values SHALL be set by the server based
 581 on the information it extracts from the certificate that is created (as a result of a Certify or a Re-certify
 582 operation) or registered (as part of a Register operation) and SHALL NOT be changed or deleted before
 583 the object is destroyed.

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

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

587 **Table 54: 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

588

Table 55: Certificate Subject Attribute Rules

589 3.11 Certificate Issuer

590 The *Certificate Issuer* attribute is a structure (see Table 57) used to identify the issuer of a certificate,
 591 containing the Issuer Distinguished Name (i.e., from the Issuer field of the certificate). It MAY include one
 592 or more alternative names (e.g., email address, IP address, DNS name) for the issuer of the certificate
 593 (i.e., from the Issuer Alternative Name extension within the certificate). The server SHALL set these
 594 values based on the information it extracts from a certificate that is created as a result of a Certify or a
 595 Re-certify operation or is sent as part of a Register operation. These values SHALL NOT be changed or
 596 deleted before the object is destroyed.

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

597

Table 56: 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

598

Table 57: Certificate Issuer Attribute Rules

599 3.12 Digest

600 The *Digest* attribute is a structure (see Table 58) that contains the digest value of the key or secret data
 601 (i.e., digest of the Key Material), certificate (i.e., digest of the Certificate Value), or opaque object (i.e.,
 602 digest of the Opaque Data Value). Multiple digests MAY be calculated using different algorithms. If an
 603 instance of this attribute exists, then it SHALL be computed with the SHA-256 hashing algorithm; the
 604 server MAY store additional digests using the algorithms listed in Section 9.1.3.2.15. The digest(s) are

605 static and SHALL be set by the server when the object is created or registered, provided that the server
 606 has access to the Key Material or the Digest Value (possibly obtained via out-of-band mechanisms).

Object	Encoding	REQUIRED
Digest	Structure	
Hashing Algorithm	Enumeration, see 9.1.3.2.15	Yes
Digest Value	Byte String	Yes, if the server has access to the Digest Value or the Key Material (for keys and secret data), the Certificate Value (for certificates) or the Opaque Data Value (for opaque objects).

607 **Table 58: Digest Attribute Structure**

SHALL always have a value	Yes, if the server has access to the Digest Value or the Key Material (for keys and secret data), the Certificate Value (for certificates) or the Opaque Data Value (for opaque objects).
Initially set by	Server
Modifiable by server	No
Modifiable by client	No
Deletable by client	No
Multiple instances permitted	Yes
When implicitly set	Create, Create Key Pair, Register, Derive Key, Certify, Re-certify, Re-key
Applies to Object Types	All Cryptographic Objects, Opaque Objects

608 **Table 59: Digest Attribute Rules**

609 3.13 Operation Policy Name

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

Object	Encoding	
Operation Policy Name	Text String	

618 **Table 60: 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

619

Table 61: Operation Policy Name Attribute Rules

620

3.13.1 Operations outside of operation policy control

621

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

622

623

- Create

624

- Create Key Pair

625

- Register

626

- Certify

627

- Re-certify

628

- Validate

629

- Query

630

- Cancel

631

- Poll

632

3.13.2 Default Operation Policy

633

A key management system implementation SHALL implement at least one named operation policy, which is used for objects when the *Operation Policy* attribute is not specified by the Client in operations that result in a new Managed Object on the server, or in a template specified in these operations. This policy is named *default*. It specifies the following rules for operations on objects created or registered with this policy, depending on the object type. For the profiles defined in **[KMIP-Prof]**, the creator SHALL be as defined in **[KMIP-Prof]**.

634

635

636

637

638

639

3.13.2.1 Default Operation Policy for Secret Objects

640

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 62: Default Operation Policy for Secret Objects

641

642 3.13.2.2 Default Operation Policy for Certificates and Public Key Objects

643 This policy applies to Certificates and Public Keys.

Default Operation Policy for Certificates and Public Key Objects	
Operation	Policy
Locate	Allowed to all
Check	Allowed to all
Get	Allowed to all
Get Attributes	Allowed to all
Get Attribute List	Allowed to all
Add Attribute	Allowed to 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

644 **Table 63: Default Operation Policy for Certificates and Public Key Objects**

645 **3.13.2.3 Default Operation Policy for Template Objects**

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

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

651 **Table 64: Default Operation Policy for Private Template Objects**

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

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

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

655 **Table 65: Default Operation Policy for Public Template Objects**

656 **3.14 Cryptographic Usage Mask**

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

- 660 • Sign
- 661 • Verify
- 662 • Encrypt
- 663 • Decrypt
- 664 • Wrap Key
- 665 • Unwrap Key
- 666 • Export
- 667 • MAC Generate
- 668 • MAC Verify
- 669 • Derive Key
- 670 • Content Commitment
- 671 • Key Agreement
- 672 • Certificate Sign
- 673 • CRL Sign
- 674 • Generate Cryptogram
- 675 • Validate Cryptogram
- 676 • Translate Encrypt
- 677 • Translate Decrypt
- 678 • Translate Wrap
- 679 • Translate Unwrap

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

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

X.509 Key Usage to Cryptographic Usage Mask Mapping	
X.509 Key Usage Value	Cryptographic Usage Mask Value
digitalSignature	Sign or Verify
contentCommitment	Content Commitment (Non Repudiation)
keyEncipherment	Wrap Key or Unwrap Key
dataEncipherment	Encrypt or Decrypt
keyAgreement	Key Agreement
keyCertSign	Certificate Sign

cRLSign	CRL Sign
encipherOnly	Encrypt
decipherOnly	Decrypt

685
686

Table 66: X.509 Key Usage to Cryptographic Usage Mask Mapping

Object	Encoding
Cryptographic Usage Mask	Integer

687

Table 67: 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

688

Table 68: Cryptographic Usage Mask Attribute Rules

689 3.15 Lease Time

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

Object	Encoding
Lease Time	Interval

697

Table 69: 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

698

Table 70: Lease Time Attribute Rules

699 **3.16 Usage Limits**

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

- 707 • *Usage Limits Total* – the total number of Usage Limits Units allowed to be protected. This is the
708 total value for the entire life of the object and SHALL NOT be changed once the object begins to
709 be used for applying cryptographic protection.
- 710 • *Usage Limits Count* – the currently remaining number of Usage Limits Units allowed to be
711 protected by the object.
- 712 • *Usage Limits Unit* – The type of quantity for which this structure specifies a usage limit (e.g., byte,
713 object).

714 When the attribute is initially set (usually during object creation or registration), the Usage Limits Count is
715 set to the Usage Limits Total value allowed for the useful life of the object, and are decremented when the
716 object is used. The server SHALL ignore the Usage Limits Count value if the attribute is specified in an
717 operation that creates a new object. Changes made via the Modify Attribute operation reflect corrections
718 to the Usage Limits Total value, but they SHALL NOT be changed once the Usage Limits Count value
719 has changed by a Get Usage Allocation operation. The Usage Limits Count value SHALL NOT be set or
720 modified by the client via the Add Attribute or Modify Attribute operations.

Object	Encoding	REQUIRED
Usage Limits	Structure	
Usage Limits Total	Long Integer	Yes
Usage Limits Count	Long Integer	Yes
Usage Limits Unit	Enumeration, see 9.1.3.2.30	Yes

721

Table 71: Usage Limits Attribute Structure

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

Table 72: Usage Limits Attribute Rules

722

723 3.17 State

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

- 729 • *Pre-Active*: The object exists but is not yet usable for
730 any cryptographic purpose.
- 731 • *Active*: The object MAY be used for all cryptographic
732 purposes that are allowed by its Cryptographic Usage
733 Mask attribute and, if applicable, by its Process Start
734 Date (see 3.20) and Protect Stop Date (see 3.21)
735 attributes.
- 736 • *Deactivated*: The object SHALL NOT be used for
737 applying cryptographic protection (e.g., encryption or
738 signing), but, if permitted by the Cryptographic Usage
739 Mask attribute, then the object MAY be used to
740 process cryptographically-protected information (e.g.,
741 decryption or verification), but only under
742 extraordinary circumstances and when special
743 permission is granted.
- 744 • *Compromised*: It is possible that the object has been
745 compromised, and SHOULD only be used to process
746 cryptographically-protected information in a client that
747 is trusted to use managed objects that have been
748 compromised.

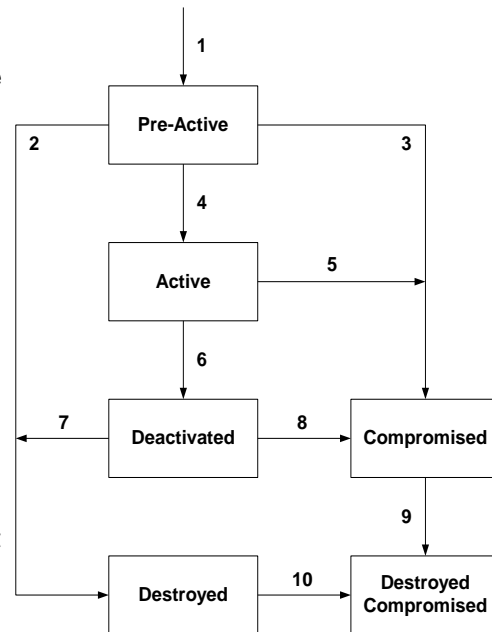


Figure 1: Cryptographic Object States and Transitions

- 749 • *Destroyed*: The object is no longer usable for any purpose.
- 750 • *Destroyed Compromised*: The object is no longer usable for any purpose; however its
- 751 compromised status MAY be retained for audit or security purposes.

752 State transitions occur as follows:

- 753 1. The transition from a non-existent key to the Pre-Active state is caused by the creation of the
754 object. When an object is created or registered, it automatically goes from non-existent to Pre-
755 Active. If, however, the operation that creates or registers the object contains an Activation Date
756 that has already occurred, then the state immediately transitions from Pre-Active to Active. In this
757 case, the server SHALL set the Activation Date attribute to the time when the operation is
758 received, or fail the request attempting to create or register the object, depending on server
759 policy. If the operation contains an Activation Date attribute that is in the future, or contains no
760 Activation Date, then the Cryptographic Object is initialized in the key management system in the
761 Pre-Active state.
- 762 2. The transition from Pre-Active to Destroyed is caused by a client issuing a Destroy operation. The
763 server destroys the object when (and if) server policy dictates.
- 764 3. The transition from Pre-Active to Compromised is caused by a client issuing a Revoke operation
765 with a Revocation Reason of Compromised.
- 766 4. The transition from Pre-Active to Active SHALL occur in one of three ways:
767 • The Activation Date is reached.
768 • A client successfully issues a Modify Attribute operation, modifying the Activation Date to a
769 date in the past, or the current date.
770 • A client issues an Activate operation on the object. The server SHALL set the Activation
771 Date to the time the Activate operation is received.
- 772 5. The transition from Active to Compromised is caused by a client issuing a Revoke operation with
773 a Revocation Reason of Compromised.
- 774 6. The transition from Active to Deactivated SHALL occur in one of three ways:
775 • The object's Deactivation Date is reached.
776 • A client issues a Revoke operation, with a Revocation Reason other than Compromised.
777 • The client successfully issues a Modify Attribute operation, modifying the Deactivation Date
778 to a date in the past, or the current date.
- 779 7. The transition from Deactivated to Destroyed is caused by a client issuing a Destroy operation, or
780 by a server, both in accordance with server policy. The server destroys the object when (and if)
781 server policy dictates.
- 782 8. The transition from Deactivated to Compromised is caused by a client issuing a Revoke operation
783 with a Revocation Reason of Compromised.
- 784 9. The transition from Compromised to Destroyed Compromised is caused by a client issuing a
785 Destroy operation, or by a server, both in accordance with server policy. The server destroys the
786 object when (and if) server policy dictates.
- 787 10. The transition from Destroyed to Destroyed Compromised is caused by a client issuing a Revoke
788 operation with a Revocation Reason of Compromised.

789 Only the transitions described above are permitted.

Object	Encoding	
State	Enumeration, see 9.1.3.2.17	

790 **Table 73: State Attribute**

SHALL always have a value	Yes
Initially set by	Server
Modifiable by server	Yes
Modifiable by client	No, but only by the server in response to certain requests (see above)
Deletable by client	No
Multiple instances permitted	No
When implicitly set	Create, Create Key Pair, Register, Derive Key, Activate, Revoke, Destroy, Certify, Re-certify, Re-key
Applies to Object Types	All Cryptographic Objects

791

Table 74: State Attribute Rules

792 **3.18 Initial Date**

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

Object	Encoding	
Initial Date	Date-Time	

798

Table 75: 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

799

Table 76: Initial Date Attribute Rules

800 **3.19 Activation Date**

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

Object	Encoding	
Activation Date	Date-Time	

805

Table 77: Activation Date Attribute

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

806

Table 78: Activation Date Attribute Rules

807 3.20 Process Start Date

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

Object	Encoding	
Process Start Date	Date-Time	

814

Table 79: Process Start Date Attribute

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

815

Table 80: Process Start Date Attribute Rules

816 **3.21 Protect Stop Date**

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

Object	Encoding	
Protect Stop Date	Date-Time	

822

Table 81: Protect Stop Date Attribute

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

823

Table 82: Protect Stop Date Attribute Rules824 **3.22 Deactivation Date**

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

Object	Encoding	
Deactivation Date	Date-Time	

830

Table 83: Deactivation Date Attribute

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

831

Table 84: Deactivation Date Attribute Rules832 **3.23 Destroy Date**

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

Object	Encoding	
Destroy Date	Date-Time	

837

Table 85: 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

838 **Table 86: Destroy Date Attribute Rules**

839 3.24 Compromise Occurrence Date

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

Object	Encoding	
Compromise Occurrence Date	Date-Time	

843 **Table 87: 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

844 **Table 88: Compromise Occurrence Date Attribute Rules**

845 3.25 Compromise Date

846 The *Compromise Date* is the date and time when the Managed Cryptographic Object entered into the
847 compromised state. This time corresponds to state transitions 3, 5, 8, or 10 (see Section 3.17). This time
848 indicates when the key management system was made aware of the compromise, not necessarily when
849 the compromise occurred. This attribute is set by the server when it receives a Revoke operation with a
850 Revocation Reason of Compromised, or due to server policy or out-of-band administrative action.

Object	Encoding	
Compromise Date	Date-Time	

851 **Table 89: 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

852 **Table 90: Compromise Date Attribute Rules**

853 3.26 Revocation Reason

854 The *Revocation Reason* attribute is a structure (see Table 91) used to indicate why the Managed
855 Cryptographic Object was revoked (e.g., “compromised”, “expired”, “no longer used”, etc). This attribute is
856 only set by the server as a part of the Revoke Operation.

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

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

860 **Table 91: 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

861 **Table 92: Revocation Reason Attribute Rules**

862 3.27 Archive Date

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

Object	Encoding	
Archive Date	Date-Time	

866

Table 93: Archive Date Attribute

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

867

Table 94: Archive Date Attribute Rules

868 3.28 Object Group

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

Object	Encoding	
Object Group	Text String	

871

Table 95: 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

872

Table 96: Object Group Attribute Rules

873 3.29 Link

874 The *Link* attribute is a structure (see Table 97) used to create a link from one Managed Cryptographic
875 Object to another, closely related target Managed Cryptographic Object. The link has a type, and the
876 allowed types differ, depending on the Object Type of the Managed Cryptographic Object, as listed
877 below. The *Linked Object Identifier* identifies the target Managed Cryptographic Object by its Unique
878 Identifier. The link contains information about the association between the Managed Cryptographic
879 Objects (e.g., the private key corresponding to a public key; the parent certificate for a certificate in a
880 chain; or for a derived symmetric key, the base key from which it was derived).

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

- 883 • *Private Key Link*. For a Public Key object: the private key corresponding to the public key.
- 884 • *Public Key Link*. For a Private Key object: the public key corresponding to the private key. For a
885 Certificate object: the public key contained in the certificate.
- 886 • *Certificate Link*. For Certificate objects: the parent certificate for a certificate in a certificate chain.
887 For Public Key objects: the corresponding certificate(s), containing the same public key.
- 888 • *Derivation Base Object Link* for a derived Symmetric Key object: the object(s) from which the
889 current symmetric key was derived.
- 890 • *Derived Key Link*: the symmetric key(s) that were derived from the current object.
- 891 • *Replacement Object Link*. For a Symmetric Key object: the key that resulted from the re-key of
892 the current key. For a Certificate object: the certificate that resulted from the re-certify. Note that
893 there SHALL be only one such replacement object per Managed Object.
- 894 • *Replaced Object Link*. For a Symmetric Key object: the key that was re-keyed to obtain the
895 current key. For a Certificate object: the certificate that was re-certified to obtain the current
896 certificate.

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

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

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

Object	Encoding	REQUIRED
Link	Structure	
Link Type	Enumeration, see 9.1.3.2.19	Yes
Linked Object Identifier, see 3.1	Text String	Yes

907 **Table 97: 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

908 **Table 98: Link Attribute Structure Rules**

909 **3.30 Application Specific Information**

910 The *Application Specific Information* attribute is a structure (see Table 99) used to store data specific to
 911 the application(s) using the Managed Object. It consists of the following fields: an *Application Namespace*
 912 and *Application Data* specific to that application namespace.

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

918

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

919 **Table 99: Application Specific Information Attribute**

920

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

921 **Table 100: Application Specific Information Attribute Rules**

922 **3.31 Contact Information**

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

Object	Encoding	
Contact Information	Text String	

925 **Table 101: 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

926 **Table 102: Contact Information Attribute Rules**

927 3.32 Last Change Date

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

Object	Encoding	
Last Change Date	Date-Time	

930 **Table 103: 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

931 **Table 104: Last Change Date Attribute Rules**

932 3.33 Custom Attribute

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

939 a prefix of 'y-'. The tag type Custom Attribute is not able to identify the particular attribute; hence such an
 940 attribute SHALL only appear in an Attribute Structure with its name as defined in Section 2.1.1.

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

941 **Table 105 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

942 **Table 106: Custom Attribute Rules**

943 4 Client-to-Server Operations

944 The following subsections describe the operations that MAY be requested by a key management client.
945 Not all clients have to be capable of issuing all operation requests; however any client that issues a
946 specific request SHALL be capable of understanding the response to the request. All Object Management
947 operations are issued in requests from clients to servers, and results obtained in responses from servers
948 to clients. Multiple operations MAY be combined within a batch, resulting in a single request/response
949 message pair.

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

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

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

965 Requests MAY contain attribute values to be assigned to the object. This information is specified with a
966 Template-Attribute (see Section 2.1.8) that contains zero or more template names and zero or more
967 individual attributes. If more than one template name is specified, and there is a conflict between the
968 single-instance attributes in the templates, then the value in the last of the conflicting templates takes
969 precedence. If there is a conflict between the single-instance attributes in the request and the single-
970 instance attributes in a specified template, then the attribute values in the request take precedence. For
971 multi-value attributes, the union of attribute values is used when the attributes are specified more than
972 once.

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

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

979 4.1 Create

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

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

985 The response contains the Unique Identifier of the created object. The server SHALL copy the Unique
986 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.

987 **Table 107: 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.

988 **Table 108: Create Response Payload**

989 Table 109 indicates which attributes SHALL be included in the Create request using the Template-
990 Attribute object.

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

991 **Table 109: Create Attribute Requirements**

992 4.2 Create Key Pair

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

995 The request contains attributes to be assigned to the objects (e.g., Cryptographic Algorithm,
996 Cryptographic Length, etc). Attributes and Template Names MAY be specified for both keys at the same
997 time by specifying a Common Template-Attribute object in the request. Attributes not common to both
998 keys (e.g., Name, Cryptographic Usage Mask) MAY be specified using the Private Key Template-Attribute
999 and Public Key Template-Attribute objects in the request, which take precedence over the Common
1000 Template-Attribute object.

1001 A Link Attribute is automatically created by the server for each object, pointing to the corresponding
1002 object. The response contains the Unique Identifiers of both created objects. The ID Placeholder value
1003 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.

1004 **Table 110: Create Key Pair Request Payload**

1005 For multi-instance attributes, the union of the values found in the templates and attributes of the
1006 Common, Private, and Public Key Template-Attribute is used. For single-instance attributes, the order of
1007 precedence is as follows:

- 1008 1. attributes specified explicitly in the Private and Public Key Template-Attribute, then
- 1009 2. attributes specified via templates in the Private and Public Key Template-Attribute, then
- 1010 3. attributes specified explicitly in the Common Template-Attribute, then
- 1011 4. attributes specified via templates in the Common Template-Attribute

1012 If there are multiple templates in the Common, Private, or Public Key Template-Attribute, then the last
1013 value of the single-instance attribute that conflicts takes precedence.

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

1014 **Table 111: Create Key Pair Response Payload**

1015 Table 112 indicates which attributes SHALL be included in the Create Key pair request using Template-
1016 Attribute objects, as well as which attributes SHALL have the same value for the Private and Public Key.

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

1017 **Table 112: Create Key Pair Attribute Requirements**

1018 Setting the same Cryptographic Length value for both private and public key does not imply that both
 1019 keys are of equal length. For RSA, Cryptographic Length corresponds to the bit length of the Modulus.
 1020 For DSA and DH algorithms, Cryptographic Length corresponds to the bit length of parameter P, and the
 1021 bit length of Q is set separately in the Cryptographic Domain Parameters attribute. For ECDSA, ECDH,
 1022 and ECMQV algorithms, Cryptographic Length corresponds to the bit length of parameter Q.

1023 **4.3 Register**

1024 This operation requests the server to register a Managed Object that was created by the client or
 1025 obtained by the client through some other means, allowing the server to manage the object. The
 1026 arguments in the request are similar to those in the Create operation, but also MAY contain the object
 1027 itself for storage by the server. Optionally, objects that are not to be stored by the key management
 1028 system MAY be omitted from the request (e.g., private keys).

1029 The request contains information about the type of object being registered and some of the attributes to
 1030 be assigned to the object (e.g., Cryptographic Algorithm, Cryptographic Length, etc). This information
 1031 MAY be specified by the use of a Template-Attribute object.

1032 The response contains the Unique Identifier assigned by the server to the registered object. The server
 1033 SHALL copy the Unique Identifier returned by this operations into the ID Placeholder variable. The Initial
 1034 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, Template 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.

1035 **Table 113: 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 114: Register Response Payload

1036

1037
1038

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

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

Table 115: Register Attribute Requirements

1039

1040

4.4 Re-key

1041
1042
1043

This request is used to generate a replacement key for an existing symmetric key. It is analogous to the Create operation, except that attributes of the replacement key are copied from the existing key, with the exception of the attributes listed in Table 117.

1044
1045

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

1046
1047

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

1048
1049

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

1050
1051
1052
1053
1054

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

Attribute in Existing Key	Attribute in Replacement Key
Initial Date (IT_1)	Initial Date (IT_2) > IT_1
Activation Date (AT_1)	Activation Date (AT_2) = IT_2 + Offset
Process Start Date (CT_1)	Process Start Date = $CT_1+(AT_2- AT_1)$
Protect Stop Date (TT_1)	Protect Stop Date = $TT_1+(AT_2- AT_1)$
Deactivation Date (DT_1)	Deactivation Date = $DT_1+(AT_2- AT_1)$

Table 116: Computing New Dates from Offset during Re-key

1055

1056

1057

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

Attribute	Action
Initial Date, see 3.18	Set to the current time
Destroy Date, see 3.23	Not set
Compromise Occurrence Date, see 3.24	Not set
Compromise Date, see 3.25	Not set
Revocation Reason, see 3.26	Not set
Unique Identifier, see 3.1	New value generated
Usage Limits, see 3.16	The Total value is copied from the existing key, and the Count value is set to the Total value.
Name, see 3.2	Set to the name(s) of the existing key; all name attributes are removed from the existing key.
State, see 3.17	Set based on attributes values, such as dates, as shown in Table 116
Digest, see 3.12	Recomputed from the replacement key value
Link, see 3.29	Set to point to the existing key as the replaced key
Last Change Date, see 3.32	Set to current time

Table 117: Re-key Attribute Requirements

1058

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the existing Symmetric Key being re-keyed. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Offset	No	An Interval object indicating the difference between the Initialization Date and the Activation Date of the replacement key to be created.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

1059 **Table 118: 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.

1060 **Table 119: Re-key Response Payload**

1061 4.5 Derive Key

1062 This request is used to derive a symmetric key or Secret Data object from a key or secret data that is
1063 already known to the key management system. The request SHALL only apply to Managed
1064 Cryptographic Objects that have the Derive Key bit set in the Cryptographic Usage Mask attribute of the
1065 specified Managed Object (i.e., are able to be used for key derivation). If the operation is issued for an
1066 object that does not have this bit set, then the server SHALL return an error. For all derivation methods,
1067 the client SHALL specify the desired length of the derived key or Secret Data object using the
1068 Cryptographic Length attribute. If a key is created, then the client SHALL specify both its Cryptographic
1069 Length and Cryptographic Algorithm. If the specified length exceeds the output of the derivation method,
1070 then the server SHALL return an error. Clients MAY derive multiple keys and IVs by requesting the
1071 creation of a Secret Data object and specifying a Cryptographic Length that is the total length of the
1072 derived object. The length SHALL NOT exceed the length of the output returned by the chosen derivation
1073 method.

1074 The fields in the request specify the Unique Identifiers of the keys or Secret Data objects to be used for
1075 derivation (e.g., some derivation methods MAY require multiple keys or Secret Data objects to derive the
1076 result), the method to be used to perform the derivation, and any parameters needed by the specified
1077 method. The method is specified as an enumerated value. Currently defined derivation methods include:

- 1078 • *PBKDF2* – This method is used to derive a symmetric key from a password or pass phrase. The
1079 *PBKDF2* method is published in **[PKCS#5]** and **[RFC2898]**.
- 1080 • *HASH* – This method derives a key by computing a hash over the derivation key or the derivation
1081 data.
- 1082 • *HMAC* – This method derives a key by computing an HMAC over the derivation data.
- 1083 • *ENCRYPT* – This method derives a key by encrypting the derivation data.

- 1084 • *NIST800-108-C* – This method derives a key by computing the KDF in Counter Mode as specified
1085 in **[SP800-108]**.
- 1086 • *NIST800-108-F* – This method derives a key by computing the KDF in Feedback Mode as
1087 specified in **[SP800-108]**.
- 1088 • *NIST800-108-DPI* – This method derives a key by computing the KDF in Double-Pipeline Iteration
1089 Mode as specified in **[SP800-108]**.
- 1090 • *Extensions*

1091 The server SHALL perform the derivation function, and then register the derived object as a new
1092 Managed Object, returning the new Unique Identifier for the new object in the response. The server
1093 SHALL copy the Unique Identifier returned by this operation into the ID Placeholder variable.

1094 As a result of Derive Key, the Link attributes (i.e., Derived Key Link in the objects from which the key is
1095 derived, and the Derivation Base Object Link in the derived key) of all objects involved SHALL be set to
1096 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 current value of the ID Placeholder SHALL NOT be used in place of a Unique Identifier in this operation.
Derivation Method, see 9.1.3.2.20	Yes	An Enumeration object specifying the method to be used to derive the new key.
Derivation Parameters, see below	Yes	A Structure object containing the parameters needed by the specified derivation method.
Template-Attribute, see 2.1.8	Yes	Specifies desired object attributes using templates and/or individual attributes; the length and algorithm SHALL always be specified for the creation of a symmetric key.

1097 **Table 120: Derive Key Request Payload**

Response Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	Yes	The Unique Identifier of the newly derived key or Secret Data object.
Template-Attribute, see 2.1.8	No	An OPTIONAL list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

1098

Table 121: Derive Key Response Payload

1099 The *Derivation Parameters* for all derivation methods consist of the following parameters, except
1100 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.

1101

Table 122: Derivation Parameters Structure (Except PBKDF2)

1102 Cryptographic Parameters identify the Pseudorandom Function (PRF) or the mode of operation of the
1103 PRF (e.g., if a key is to be derived using the HASH derivation method, then clients are REQUIRED to
1104 indicate the hash algorithm inside Cryptographic Parameters; similarly, if a key is to be derived using AES
1105 in CBC mode, then clients are REQUIRED to indicate the Block Cipher Mode). The server SHALL verify
1106 that the specified mode matches one of the instances of Cryptographic Parameters set for the
1107 corresponding key. If Cryptographic Parameters are omitted, then the server SHALL select the
1108 Cryptographic Parameters with the lowest Attribute Index for the specified key. If the corresponding key
1109 does not have any Cryptographic Parameters attribute, or if no match is found, then an error is returned.

1110 If a key is derived using HMAC, then the attributes of the derivation key provide enough information about
1111 the PRF and the Cryptographic Parameters are ignored.

1112 Derivation Data is either the data to be encrypted, hashed, or HMACed. For the NIST SP 800-108
1113 methods **[SP800-108]**, Derivation Data is Label||{(0x00)}||Context, where the all-zero byte is OPTIONAL.

1114 Most derivation methods (e.g., ENCRYPT) require a derivation key and the derivation data to be used.
1115 The HASH derivation method requires either a derivation key or derivation data. Derivation data MAY
1116 either be explicitly provided by the client with the Derivation Data field or implicitly provided by providing
1117 the Unique Identifier of a Secret Data object. If both are provided, then an error SHALL be returned.

1118 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 (if

		different than those defined in [PKCS#5]) and mode of operation: an empty IV is assumed if not provided.
Derivation Data	Byte String	Yes, unless the Unique Identifier of a Secret Data object is provided.
Salt	Byte String	Yes
Iteration Count	Integer	Yes

1119

Table 123: PBKDF2 Derivation Parameters Structure

1120

4.6 Certify

1121 This request is used to generate a Certificate object for a public key. This request supports certification of
 1122 a new public key as well as certification of a public key that has already been certified (i.e., certificate
 1123 update). Only a single certificate SHALL be requested at a time. Server support for this operation is
 1124 OPTIONAL, as it requires that the key management system have access to a certification authority (CA).
 1125 If the server does not support this operation, an error SHALL be returned.

1126 The Certificate Request is passed as a Byte String, which allows multiple certificate request types for
 1127 X.509 certificates (e.g., PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

1128 The generated Certificate object whose Unique Identifier is returned MAY be obtained by the client via a
 1129 Get operation in the same batch, using the ID Placeholder mechanism.

1130 As a result of Certify, the Link attribute of the Public Key and of the generated certificate SHALL be set to
 1131 point at each other.

1132 The server SHALL copy the Unique Identifier of the generated certificate returned by this operation into
 1133 the ID Placeholder variable.

1134 If the information in the Certificate Request conflicts with the attributes specified in the Template-Attribute,
 1135 then the information in the Certificate Request takes precedence.

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	The Unique Identifier of the Public Key being certified. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Certificate Request Type, see 9.1.3.2.21	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.

1136

Table 124: 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.

1137

Table 125: Certify Response Payload

1138 4.7 Re-certify

1139 This request is used to renew an existing certificate for the same key pair. Only a single certificate SHALL
 1140 be renewed at a time. Server support for this operation is OPTIONAL, as it requires that the key
 1141 management system to have access to a certification authority (CA). If the server does not support this
 1142 operation, an error SHALL be returned.

1143 The Certificate Request is passed as a Byte String, which allows multiple certificate request types for
 1144 X.509 certificates (e.g., PKCS#10, PEM, etc) or PGP certificates to be submitted to the server.

1145 The server SHALL copy the Unique Identifier of the new certificate returned by this operation into the ID
 1146 Placeholder variable.

1147 If the information in the Certificate Request field in the request conflicts with the attributes specified in the
 1148 Template-Attribute, then the information in the Certificate Request takes precedence.

1149 As the new certificate takes over the name attribute of the existing certificate, Re-certify SHOULD only be
 1150 performed once on a given (existing) certificate.

1151 The Link attribute of the existing certificate and of the new certificate are set to point at each other. The
 1152 Link attribute of the Public Key is changed to point to the new certificate.

1153 An *Offset* MAY be used to indicate the difference between the Initialization Date and the Activation Date
 1154 of the new certificate. If Offset is set, then the dates of the new certificate SHALL be set based on the
 1155 dates of the existing certificate (if such dates exist) as follows:

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

1156

Table 126: Computing New Dates from Offset during Re-certify

1157 Attributes that are not copied from the existing certificate and that are handled in a specific way for the
 1158 new certificate are:

Attribute	Action
Initial Date, see 3.18	Set to current time
Destroy Date, see 3.23	Not set
Revocation Reason, see 3.26	Not set
Unique Identifier, see 3.2	New value generated
Name, see 3.2	Set to the name(s) of the existing certificate; all name attributes are removed from the existing certificate.
State, see 3.17	Set based on attributes values, such as dates, as shown in Table 126
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

1159

Table 127: Re-certify Attribute Requirements

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	The Unique Identifier of the Certificate being renewed. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Certificate Request Type, see 9.1.3.2.21	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 Initial Date of the new certificate and the Activation Date of the new certificate.
Template-Attribute, see 2.1.8	No	Specifies desired object attributes using templates and/or individual attributes.

1160

Table 128: 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.

1161

Table 129: Re-certify Response Payload

1162 4.8 Locate

1163 This operation requests that the server search for one or more Managed Objects depending on the
 1164 attributes specified in the request. All attributes are allowed to be used. However, Attribute Index values
 1165 SHOULD NOT be specified in the request. Attribute Index values that are provided SHALL be ignored by
 1166 the Locate operation. The request MAY also contain a *Maximum Items* field, which specifies the
 1167 maximum number of objects to be returned. If the Maximum Items field is omitted, then the server MAY
 1168 return all objects matched, or MAY impose an internal maximum limit due to resource limitations.

1169 If more than one object satisfies the identification criteria specified in the request, then the response MAY
 1170 contain Unique Identifiers for multiple Managed Objects. Returned objects SHALL match **all** of the
 1171 attributes in the request. If no objects match, then an empty response payload is returned. If no attribute
 1172 is specified in the request, any object SHALL be deemed to match the Locate request.

1173 The server returns a list of Unique Identifiers of the found objects, which then MAY be retrieved using the
 1174 Get operation. If the objects are archived, then the Recover and Get operations are REQUIRED to be
 1175 used to obtain those objects. If a single Unique Identifier is returned to the client, then the server SHALL
 1176 copy the Unique Identifier returned by this operation into the ID Placeholder variable. If the Locate
 1177 operation matches more than one object, and the Maximum Items value is omitted in the request, or is set
 1178 to a value larger than one, then the server SHALL empty the ID Placeholder, causing any subsequent
 1179 operations that are batched with the Locate, and which do not specify a Unique Identifier explicitly, to fail.
 1180 This ensures that these batched operations SHALL proceed only if a single object is returned by Locate.

1181 Wild-cards or regular expressions (defined, e.g., in [ISO/IEC 9945-2]) MAY be supported by specific key
 1182 management system implementations for matching attribute fields when the field type is a Text String or a
 1183 Byte String.

1184 The Date attributes in the Locate request (e.g., Initial Date, Activation Date, etc) are used to specify a
 1185 time or a time range for the search. If a single instance of a given Date attribute is used in the request
 1186 (e.g., the Activation Date), then objects with the same Date attribute are considered to be matching
 1187 candidate objects. If two instances of the same Date attribute are used (i.e., with two different values
 1188 specifying a range), then objects for which the Date attribute is inside or at a limit of the range are
 1189 considered to be matching candidate objects. If a Date attribute is set to its largest possible value, then it
 1190 is equivalent to an undefined attribute. The KMIP Usage Guide [KMIP-UG] provides examples.

1191 When the Cryptographic Usage Mask attribute is specified in the request, candidate objects are
 1192 compared against this field via an operation that consists of a logical AND of the requested mask with the
 1193 mask in the candidate object, and then a comparison of the resulting value with the requested mask. For
 1194 example, if the request contains a mask value of 10001100010000, and a candidate object mask contains
 1195 10000100010000, then the logical AND of the two masks is 10000100010000, which is compared against
 1196 the mask value in the request (10001100010000) and the match fails. This means that a matching
 1197 candidate object has all of the bits set in its mask that are set in the requested mask, but MAY have
 1198 additional bits set.

1199 When the Usage Allocation attribute is specified in the request, matching candidate objects SHALL have
 1200 an Object or Byte Count and Total Objects or Bytes equal to or larger than the values specified in the
 1201 request.

1202 When an attribute that is defined as a structure is specified, all of the structure fields are not REQUIRED
 1203 to be specified. For instance, for the Link attribute, if the Linked Object Identifier value is specified without
 1204 the Link Type value, then matching candidate objects have the Linked Object Identifier as specified,
 1205 irrespective of their Link Type.

1206 The Storage Status Mask field (see Section 9.1.3.3.2) is used to indicate whether only on-line objects,
 1207 only archived objects, or both on-line and archived objects are to be searched. Note that the server MAY
 1208 store attributes of archived objects in order to expedite Locate operations that search through archived
 1209 objects.

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

1210 **Table 130: Locate Request Payload**

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

1211 **Table 131: Locate Response Payload**

1212 4.9 Check

1213 This operation requests that the server check for the use of a Managed Object according to values
 1214 specified in the request. This operation SHOULD only be used when placed in a batched set of
 1215 operations, usually following a Locate, Create, Create Pair, Derive Key, Certify, Re-Certify or Re-Key
 1216 operation, and followed by a Get operation.

1217 If the server determines that the client is allowed to use the object according to the specified attributes,
 1218 then the server returns the Unique Identifier of the object.

1219 If the server determines that the client is not allowed to use the object according to the specified
 1220 attributes, then the server empties the ID Placeholder and does not return the Unique Identifier, and the
 1221 operation returns the set of attributes specified in the request that caused the server policy denial. The
 1222 only attributes returned are those that resulted in the server determining that the client is not allowed to
 1223 use the object, thus allowing the client to determine how to proceed. The operation also returns a failure,
 1224 and the server SHALL ignore any subsequent operations in the batch.

1225 The additional objects that MAY be specified in the request are limited to:

- 1226 • Usage Limits Count (see Section 3.16) – The request MAY contain the usage amount that the
 1227 client deems necessary to complete its needed function. This does not require that any

1228 subsequent Get Usage Allocation operations request this amount. It only means that the client is
 1229 ensuring that the amount specified is available.

- 1230 • Cryptographic Usage Mask – This is used to specify the cryptographic operations for which the
 1231 client intends to use the object (see Section 3.14). This allows the server to determine if the policy
 1232 allows this client to perform these operations with the object. Note that this MAY be a different
 1233 value from the one specified in a Locate operation that precedes this operation. Locate, for
 1234 example, MAY specify a Cryptographic Usage Mask requesting a key that MAY be used for both
 1235 Encryption and Decryption, but the value in the Check operation MAY specify that the client is
 1236 only using the key for Encryption at this time.
- 1237 • Lease Time – This specifies a desired lease time (see Section 3.15). The client MAY use this to
 1238 determine if the server allows the client to use the object with the specified lease or longer.
 1239 Including this attribute in the Check operation does not actually cause the server to grant a lease,
 1240 but only indicates that the requested lease time value MAY be granted if requested by a
 1241 subsequent, batched, Obtain Lease operation.

1242 Note that these objects are not encoded in an Attribute structure as shown in Section 2.1.1

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the object being checked. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Usage Limits Count, see 3.16	No	Specifies the number of Usage Limits Units to be protected to be checked against server policy.
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.

1243 **Table 132: Check Request Payload**

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

1244 **Table 133: Check Response Payload**

1245 **4.10 Get**

1246 This operation requests that the server returns the Managed Object specified by its Unique Identifier.

1247 Only a single object is returned. The response contains the Unique Identifier of the object, along with the
 1248 object itself, which MAY be wrapped using a wrapping key as specified in the request.

1249 The following key format capabilities SHALL be assumed by the client restrictions apply when the client
 1250 requests the server to return an object in a particular format:

- 1251 • If a client registered a key in a given format, the server SHALL be able to return the key during
 1252 the Get operation in the same format that was used when the key was registered.
- 1253 • Any other format conversion MAY optionally be supported by the server.

1254

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the object being requested. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Key Format Type, see 9.1.3.2.3	No	Determines the key format type to be returned.
Key Compression Type, see 9.1.3.2.2	No	Determines the compression method for elliptic curve public keys.
Key Wrapping Specification, see 2.1.6	No	Specifies keys and other information for wrapping the returned object. This field SHALL NOT be specified if the requested object is a Template.

1255

Table 134: 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.

1256

Table 135: Get Response Payload

1257 **4.11 Get Attributes**

1258 This operation requests one or more attributes of a Managed Object. The object is specified by its Unique
 1259 Identifier and the attributes are specified by their name in the request. If a specified attribute has multiple
 1260 instances, then all instances are returned. If a specified attribute does not exist (i.e., has no value), then it
 1261 SHALL NOT be present in the returned response. If no requested attributes exist, then the response
 1262 SHALL consist only of the Unique Identifier. If no attribute name is specified in the request, all attributes
 1263 SHALL be deemed to match the Get Attributes request.

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the object whose attributes are being requested. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Attribute Name, see 2.1.1	No, MAY be repeated	Specifies a desired attribute of the object.

1264

Table 136: 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.

1265

Table 137: Get Attributes Response Payload

1266 4.12 Get Attribute List

1267 This operation requests a list of the attribute names associated with a Managed Object. The object is
1268 specified by its Unique Identifier.

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the object whose attribute names are being requested. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.

1269

Table 138: Get Attribute List Request Payload

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

1270

Table 139: Get Attribute List Response Payload

1271 4.13 Add Attribute

1272 This request adds a new attribute instance to a Managed Object and sets its value. The request contains
1273 the Unique Identifier of the Managed Object to which the attribute pertains, along with the attribute name
1274 and value. For non-multi-instance attributes, this is how the attribute value is created. For multi-instance
1275 attributes, this is how the first and subsequent values are created. Existing attribute values SHALL only
1276 be changed by the Modify Attribute operation. Read-Only attributes SHALL NOT be added using the Add
1277 Attribute operation. No Attribute Index SHALL be specified in the request. The response returns a new
1278 Attribute Index, although the Attribute Index MAY be omitted if the index of the added attribute instance is
1279 0. Multiple Add Attribute requests MAY be included in a single batched request to add multiple attributes.

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

1280

Table 140: 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.

1281

Table 141: Add Attribute Response Payload

1282 4.14 Modify Attribute

1283 This request modifies the value of an existing attribute instance associated with a Managed Object. The
 1284 request contains the Unique Identifier of the Managed Object whose attribute is to be modified, and the
 1285 attribute name, OPTIONAL Attribute Index, and the new value. Only existing attributes MAY be changed
 1286 via this operation. New attributes SHALL only be added by the Add Attribute operation. If an Attribute
 1287 Index is specified, then only the specified instance of the attribute is modified. If the attribute has multiple
 1288 instances, and no Attribute Index is specified in the request, then the Attribute Index is assumed to be 0.
 1289 If the attribute does not support multiple instances, then the Attribute Index SHALL NOT be specified.
 1290 Specifying an Attribute Index for which there exists no 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 value is used by the server as the Unique Identifier.
Attribute, see 2.1.1	Yes	Specifies the attribute of the object to be modified.

1291

Table 142: 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 with the new value.

1292

Table 143: Modify Attribute Response Payload

1293 4.15 Delete Attribute

1294 This request deletes an attribute associated with a Managed Object. The request contains the Unique
 1295 Identifier of the Managed Object whose attribute is to be deleted, the attribute name, and optionally the

1296 Attribute Index of the attribute. Attributes that are always required to have a value SHALL never be
 1297 deleted by this operation. If no Attribute Index is specified, and the Attribute whose name is specified has
 1298 multiple instances, then the operation is rejected. Note that only a single attribute instance SHALL be
 1299 deleted at a time. Multiple delete operations (e.g., possibly batched) are necessary to delete several
 1300 attribute instances. Attempting to delete a non-existent attribute or specifying an Attribute Index for which
 1301 there exists no Attribute Value 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 value is used by the server as the Unique Identifier.
Attribute Name, see 2.1.1	Yes	Specifies the name of the attribute to be deleted.
Attribute Index, see 2.1.1	No	Specifies the Index of the Attribute.

1302 **Table 144: 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.

1303 **Table 145: Delete Attribute Response Payload**

1304 **4.16 Obtain Lease**

1305 This request is used to obtain a new *Lease Time* for a specified Managed Object. The Lease Time is an
 1306 interval value that determines when the client's internal cache of information about the object expires and
 1307 needs to be renewed. If the returned value of the lease time is zero, then the server is indicating that no
 1308 lease interval is effective, and the client MAY use the object without any lease time limit. If a client's lease
 1309 expires, then the client SHALL NOT use the associated cryptographic object until a new lease is
 1310 obtained. If the server determines that a new lease SHALL NOT be issued for the specified cryptographic
 1311 object, then the server SHALL respond to the Obtain Lease request with an error.

1312 The response payload for the operation contains the current value of the Last Change Date attribute for
 1313 the object. This MAY be used by the client to determine if any of the attributes cached by the client need
 1314 to be refreshed, by comparing this time to the time when the attributes were previously obtained.

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

1315 **Table 146: 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.

1316

Table 147: Obtain Lease Response Payload

1317

4.17 Get Usage Allocation

1318

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

1329

The field in the request specifies the number of units that the client needs to protect. If the requested amount is not available or if the Managed Object is not able to be used for applying cryptographic protection at this time, then the server SHALL return an error. The server SHALL assume that the entire allocated amount is going to be consumed. Once the entire allocated amount has been consumed, the client SHALL NOT continue to use the Managed Cryptographic Object for applying cryptographic protection until a new allocation is obtained.

1330

1331

1332

1333

1334

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the object whose usage allocation is being requested. If omitted, then the ID Placeholder is substituted by the server.
Usage Limits Count, see Usage Limits Count field in 3.16	Yes	The number of Usage Limits Units to be protected.

1335

Table 148: Get Usage Allocation Request Payload

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

1336

Table 149: Get Usage Allocation Response Payload

1337 **4.18 Activate**

1338 This request is used to activate a Managed Cryptographic Object. The request SHALL NOT specify a
 1339 Template object. The operation SHALL only be performed on an object in the Pre-Active state and has
 1340 the effect of changing its state to Active, and setting its Activation Date to the current date and time.

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

1341 **Table 150: Activate Request Payload**

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

1342 **Table 151: Activate Response Payload**

1343 **4.19 Revoke**

1344 This request is used to revoke a Managed Cryptographic Object or an Opaque Object. The request
 1345 SHALL NOT specify a Template object. The request contains a reason for the revocation (e.g., “key
 1346 compromise”, “cessation of operation”, etc). Special authentication and authorization SHOULD be
 1347 enforced to perform this request (see [KMIP-UG]). Only the object creator or an authorized security
 1348 officer SHOULD be allowed to issue this request. The operation has one of two effects. If the revocation
 1349 reason is “key compromise”, then the object is placed into the “compromised” state, and the Compromise
 1350 Date attribute is set to the current date and time. Otherwise, the object is placed into the “deactivated”
 1351 state, and the Deactivation Date attribute is set to the current date and time.

Request Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	No	Determines the object being revoked. If omitted, then the ID Placeholder value is used by the server as the Unique Identifier.
Revocation Reason, see 3.26	Yes	Specifies the reason for revocation.
Compromise Occurrence Date, see 3.24	No	SHALL be specified if the Revocation Reason is 'compromised'.

1352 **Table 152: Revoke Request Payload**

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

1353 **Table 153: Revoke Response Payload**

1354 **4.20 Destroy**

1355 This request is used to indicate to the server that the key material for the specified Managed Object
 1356 SHALL be destroyed. The meta-data for the key material MAY be retained by the server (e.g., used to

1357 ensure that an expired or revoked private signing key is no longer available). Special authentication and
 1358 authorization SHOULD be enforced to perform this request (see [KMIP-UG]). Only the object creator or
 1359 an authorized security officer SHOULD be allowed to issue this request. If the Unique Identifier specifies
 1360 a Template object, then the object itself, including all meta-data, SHALL be destroyed. Cryptographic
 1361 Objects MAY only be destroyed if they are in either Pre-Active or Deactivated state. A Cryptographic
 1362 Object in the Active state MAY be destroyed if the server sets the Deactivation date (the state of the
 1363 object transitions to Deactivated) prior to destroying the object.

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

1364 **Table 154: Destroy Request Payload**

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

1365 **Table 155: Destroy Response Payload**

1366 4.21 Archive

1367 This request is used to specify that a Managed Object MAY be archived. The actual time when the object
 1368 is archived, the location of the archive, or level of archive hierarchy is determined by the policies within
 1369 the key management system and is not specified by the client. The request contains the unique identifier
 1370 of the Managed Object. Special authentication and authorization SHOULD be enforced to perform this
 1371 request (see [KMIP-UG]). Only the object creator or an authorized security officer SHOULD be allowed to
 1372 issue this request. This request is only an indication from a client that from its point of view it is possible
 1373 for the key management system to archive the object.

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

1374 **Table 156: Archive Request Payload**

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

1375 **Table 157: Archive Response Payload**

1376 4.22 Recover

1377 This request is used to obtain access to a Managed Object that has been archived. This request MAY
 1378 require asynchronous polling to obtain the response due to delays caused by retrieving the object from
 1379 the archive. Once the response is received, the object is now on-line, and MAY be obtained (e.g., via a
 1380 Get operation). Special authentication and authorization SHOULD be enforced to perform this request
 1381 (see [KMIP-UG]).

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

1382

Table 158: Recover Request Payload

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

1383

Table 159: Recover Response Payload

1384 4.23 Validate

1385 This requests that the server validate a certificate chain and return information on its validity. Only a
 1386 single certificate chain SHALL be included in each request. Support for this operation at the server is
 1387 OPTIONAL. If the server does not support this operation, an error SHALL be returned.

1388 The request may contain a list of certificate objects, and/or a list of Unique Identifiers that identify
 1389 Managed Certificate objects. Together, the two lists compose a certificate chain to be validated. The
 1390 request MAY also contain a date for which all certificates in the certificate chain are REQUIRED to be
 1391 valid.

1392 The method or policy by which validation is conducted is a decision of the server and is outside of the
 1393 scope of this protocol. Likewise, the order in which the supplied certificate chain is validated and the
 1394 specification of trust anchors used to terminate validation are also controlled by the server.

Request Payload		
Object	REQUIRED	Description
Certificate, see 2.2.1	No, MAY be repeated	One or more Certificates.
Unique Identifier, see 3.1	No, MAY be repeated	One or more Unique Identifiers of Certificate Objects.
Validity Date	No	A Date-Time object indicating when the certificate chain needs to be valid. If omitted, the current date and time SHALL be assumed.

1395

Table 160: 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.

1396

Table 161: Validate Response Payload

1397 **4.24 Query**

1398 This request is used by the client to interrogate the server to determine its capabilities and/or protocol
 1399 mechanisms. The *Query* operation SHOULD be invocable by unauthenticated clients to interrogate server
 1400 features and functions. The *Query Function* field in the request SHALL contain one or more of the
 1401 following items:

- 1402 • Query Operations
- 1403 • Query Objects
- 1404 • Query Server Information
- 1405 • Query Application Namespaces

1406 The *Operation* fields in the response contain Operation enumerated values, which SHALL list all the
 1407 operations that the server supports. If the request contains a Query Operations value in the Query
 1408 Function field, then these fields SHALL be returned in the response.

1409 The *Object Type* fields in the response contain Object Type enumerated values, which SHALL list all the
 1410 object types that the server supports. If the request contains a *Query Objects* value in the Query Function
 1411 field, then these fields SHALL be returned in the response.

1412 The *Server Information* field in the response is a structure containing vendor-specific fields and/or
 1413 substructures. If the request contains a *Query Server Information* value in the Query Function field, then
 1414 this field SHALL be returned in the response.

1415 The Application Namespace fields in the response contain the namespaces that the server SHALL
 1416 generate values for if requested by the client (see Section 3.30). These fields SHALL only be returned in
 1417 the response if the request contains a Query Application Namespaces value in the Query Function field.

1418 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

1419 **Table 162: 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.
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.

1420 **Table 163: Query Response Payload**

1421 **4.25 Cancel**

1422 This request is used to cancel an outstanding asynchronous operation. The correlation value (see Section
 1423 6.8) of the original operation SHALL be specified in the request. The server SHALL respond with a
 1424 *Cancellation Result* that contains one of the following values:

- 1425 • *Canceled* – The cancel operation succeeded in canceling the pending operation.
- 1426 • *Unable To Cancel* – The cancel operation is unable to cancel the pending operation.
- 1427 • *Completed* – The pending operation completed successfully before the cancellation operation
 1428 was able to cancel it.
- 1429 • *Failed* – The pending operation completed with a failure before the cancellation operation was
 1430 able to cancel it.
- 1431 • *Unavailable* – The specified correlation value did not match any recently pending or completed
 1432 asynchronous operations.

1433 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.

1434 **Table 164: Cancel Request Payload**

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

1435 **Table 165: Cancel Response Payload**

1436 **4.26 Poll**

1437 This request is used to poll the server in order to obtain the status of an outstanding asynchronous
 1438 operation. The correlation value (see Section 6.8) of the original operation SHALL be specified in the
 1439 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.

1440 **Table 166: Poll Request Payload**

1441 The server SHALL reply with one of two responses:

1442 If the operation has not completed, the response SHALL contain no payload and a Result Status of
 1443 Pending.

1444 If the operation has completed, the response SHALL contain the appropriate payload for the operation.
 1445 This response SHALL be identical to the response that would have been sent if the operation had
 1446 completed synchronously.

1447 5 Server-to-Client Operations

1448 Server-to-client operations are used by servers to send information or Managed Cryptographic Objects to
1449 clients via means outside of the normal client-server request-response mechanism. These operations are
1450 used to send Managed Cryptographic Objects directly to clients without a specific request from the client.

1451 5.1 Notify

1452 This operation is used to notify a client of events that resulted in changes to attributes of an object. This
1453 operation is only ever sent by a server to a client via means outside of the normal client request/response
1454 protocol, using information known to the server via unspecified configuration or administrative
1455 mechanisms. It contains the Unique Identifier of the object to which the notification applies, and a list of
1456 the attributes whose changed values have triggered the notification. The message uses the same format
1457 as a Request message (see 7.1, Table 185), except that the Maximum Response Size, Asynchronous
1458 Indicator, Batch Error Continuation Option, and Batch Order Option fields are not allowed. The client
1459 SHALL send a response in the form of a Response Message (see 7.1, Table 186) containing no payload,
1460 unless both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the
1461 client is not able to respond.

Message Payload		
Object	REQUIRED	Description
Unique Identifier, see 3.1	Yes	The Unique Identifier of the object.
Attribute, see 3	Yes, MAY be repeated	The attributes that have changed. This includes at least the Last Change Date attribute. In case an attribute was deleted, the Attribute structure (see 2.1.1) in question SHALL NOT contain the Attribute Value field.

1462 **Table 167: Notify Message Payload**

1463 5.2 Put

1464 This operation is used to “push” Managed Cryptographic Objects to clients. This operation is only ever
1465 sent by a server to a client via means outside of the normal client request/response protocol, using
1466 information known to the server via unspecified configuration or administrative mechanisms. It contains
1467 the Unique Identifier of the object that is being sent, and the object itself. The message uses the same
1468 format as a Request message (see 7.1, Table 185), except that the Maximum Response Size,
1469 Asynchronous Indicator, Batch Error Continuation Option, and Batch Order Option fields are not allowed.
1470 The client SHALL send a response in the form of a Response Message (see 7.1, Table 186) containing
1471 no payload, unless both the client and server have prior knowledge (obtained via out-of-band
1472 mechanisms) that the client is not able to respond.

1473 The *Put Function* field indicates whether the object being “pushed” is a new object, or is a replacement for
1474 an object already known to the client (e.g., when pushing a certificate to replace one that is about to
1475 expire, the Put Function field would be set to indicate replacement, and the Unique Identifier of the
1476 expiring certificate would be placed in the *Replaced Unique Identifier* field). The Put Function SHALL
1477 contain one of the following values:

- 1478 • *New* – which indicates that the object is not a replacement for another object.
- 1479 • *Replace* – which indicates that the object is a replacement for another object, and that the
1480 Replaced Unique Identifier field is present and contains the identification of the replaced object. In
1481 case the object with the Replaced Unique Identifier does not exist at the client, the client SHALL
1482 interpret this as if the Put Function contained the value *New*.

1483 The Attribute field contains one or more attributes that the server is sending along with the object. The
 1484 server MAY include attributes with the object to specify how the object is to be used by the client. The
 1485 server MAY include a Lease Time attribute that grants a lease to the client.

1486 If the Managed Object is a wrapped key, then the key wrapping specification SHALL be exchanged prior
 1487 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.

1488

Table 168: Put Message Payload

1489 6 Message Contents

1490 The messages in the protocol consist of a message header, one or more batch items (which contain
1491 OPTIONAL message payloads), and OPTIONAL message extensions. The message headers contain
1492 fields whose presence is determined by the protocol features used (e.g., asynchronous responses). The
1493 field contents are also determined by whether the message is a request or a response. The message
1494 payload is determined by the specific operation being requested or to which is being replied.

1495 The message headers are structures that contain some of the following objects.

1496 6.1 Protocol Version

1497 This field contains the version number of the protocol, ensuring that the protocol is fully understood by
1498 both communicating parties. The version number SHALL be specified in two parts, major and minor.
1499 Servers and clients SHALL support backward compatibility with versions of the protocol with the same
1500 major version. Support for backward compatibility with different major versions is OPTIONAL.

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

1501 **Table 169: Protocol Version Structure in Message Header**

1502 6.2 Operation

1503 This field indicates the operation being requested or the operation for which the response is being
1504 returned. The operations are defined in Sections 4 and 5

Object	Encoding
Operation	Enumeration, see 9.1.3.2.26

1505 **Table 170: Operation in Batch Item**

1506 6.3 Maximum Response Size

1507 This field is optionally contained in a request message, and is used to indicate the maximum size of a
1508 response, in bytes, that the requester SHALL handle. It SHOULD only be sent in requests that possibly
1509 return large replies.

Object	Encoding
Maximum Response Size	Integer

1510 **Table 171: Maximum Response Size in Message Request Header**

1511 6.4 Unique Batch Item ID

1512 This field is optionally contained in a request, and is used for correlation between requests and
1513 responses. If a request has a *Unique Batch Item ID*, then responses to that request SHALL have the
1514 same Unique Batch Item ID.

Object	Encoding
Unique Batch Item ID	Byte String

1515

Table 172: Unique Batch Item ID in Batch Item

1516 6.5 Time Stamp

1517 This field is optionally contained in a client request. It is REQUIRED in a server request and response. It
1518 is used for time stamping, and MAY be used to enforce reasonable time usage at a client (e.g., a server
1519 MAY choose to reject a request if a client's time stamp contains a value that is too far off the server's
1520 time). Note that the time stamp MAY be used by a client that has no real-time clock, but has a countdown
1521 timer, to obtain useful "seconds from now" values from all of the Date attributes by performing a
1522 subtraction.

Object	Encoding
Time Stamp	Date-Time

1523

Table 173: Time Stamp in Message Header

1524 6.6 Authentication

1525 This is used to authenticate the requester. It is an OPTIONAL information item, depending on the type of
1526 request being issued and on server policies. Servers MAY require authentication on no requests, a
1527 subset of the requests, or all requests, depending on policy. Query operations used to interrogate server
1528 features and functions SHOULD NOT require authentication. The Authentication structure SHALL contain
1529 a Credential structure.

1530 The authentication mechanisms are described and discussed in Section 8.

Object	Encoding
Authentication	Structure
Credential	Structure, see 2.1.2

1531

Table 174: Authentication Structure in Message Header

1532 6.7 Asynchronous Indicator

1533 This Boolean flag indicates whether the client is able to accept an asynchronous response. It SHALL
1534 have the Boolean value True if the client is able to handle asynchronous responses, and the value False
1535 otherwise. If not present in a request, then False is assumed. If a client indicates that it is not able to
1536 handle asynchronous responses (i.e., flag is set to False), and the server is not able to process the
1537 request synchronously, then the server SHALL respond to the request with a failure.

Object	Encoding
Asynchronous Indicator	Boolean

1538

Table 175: Asynchronous Indicator in Message Request Header

1539 6.8 Asynchronous Correlation Value

1540 This is returned in the immediate response to an operation that is pending and that requires
1541 asynchronous polling. Note: the server decides which operations are performed synchronously or
1542 asynchronously. A server-generated correlation value SHALL be specified in any subsequent Poll or
1543 Cancel operations that pertain to the original operation.

Object	Encoding
Asynchronous Correlation Value	Byte String

1544

Table 176: Asynchronous Correlation Value in Response Batch Item

1545 6.9 Result Status

1546 This is sent in a response message and indicates the success or failure of a request. The following values
1547 MAY be set in this field:

- 1548 • *Success* – The requested operation completed successfully.
- 1549 • *Operation Pending* – The requested operation is in progress, and it is necessary to obtain the
1550 actual result via asynchronous polling. The asynchronous correlation value SHALL be used for
1551 the subsequent polling of the result status.
- 1552 • *Operation Undone* – The requested operation was performed, but had to be undone (i.e., due to a
1553 failure in a batch for which the Error Continuation Option (see 6.13 and 7.2) was set to Undo).
- 1554 • *Operation Failed* – The requested operation failed.

Object	Encoding
Result Status	Enumeration, see 9.1.3.2.27

1555 **Table 177: Result Status in Response Batch Item**

1556 6.10 Result Reason

1557 This field indicates a reason for failure or a modifier for a partially successful operation and SHALL be
1558 present in responses that return a Result Status of Failure. In such a case, the Result Reason SHALL be
1559 set as specified in Section 11. It is OPTIONAL in any response that returns a Result Status of Success.
1560 The following defined values are defined for this field:

- 1561 • *Item not found* – A requested object was not found or did not exist.
- 1562 • *Response too large* – The response to a request would exceed the *Maximum Response Size* in
1563 the request.
- 1564 • *Authentication not successful* – The authentication information in the request was not able to be
1565 validated, or there was no authentication information in the request when there SHOULD have
1566 been.
- 1567 • *Invalid message* – The request message was not understood by the server.
- 1568 • *Operation not supported* – The operation requested by the request message is not supported by
1569 the server.
- 1570 • *Missing data* – The operation requires additional OPTIONAL information in the request, which
1571 was not present.
- 1572 • *Invalid field* – Some data item in the request has an invalid value.
- 1573 • *Feature not supported* – An OPTIONAL feature specified in the request is not supported.
- 1574 • *Operation canceled by requester* – The operation was asynchronous, and the operation was
1575 canceled by the Cancel operation before it completed successfully.
- 1576 • *Cryptographic failure* – The operation failed due to a cryptographic error.
- 1577 • *Illegal operation* – The client requested an operation that was not able to be performed with the
1578 specified parameters.
- 1579 • *Permission denied* – The client does not have permission to perform the requested operation.
- 1580 • *Object archived* – The object SHALL be recovered from the archive before performing the
1581 operation.
- 1582 • *Index Out of Bounds* – The client tried to set more instances than the server supports of an
1583 attribute that MAY have multiple instances.

- 1584 • *Application Namespace Not Supported* – The particular Application Namespace is not supported,
1585 and server was not able to generate the Application Data field of an Application Specific
1586 Information attribute if the field was omitted from the client request.
- 1587 • *Key Format Type and/or Key Compression Type Not Supported* – The object exists but the server
1588 is unable to provide it in the desired Key Format Type and/or Key Compression Type.
- 1589 • *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

1590 **Table 178: Result Reason in Response Batch Item**

1591 6.11 Result Message

1592 This field MAY be returned in a response. It contains a more descriptive error message, which MAY be
1593 provided to an end user or used for logging/auditing purposes.

Object	Encoding
Result Message	Text String

1594 **Table 179: Result Message in Response Batch Item**

1595 6.12 Batch Order Option

1596 A Boolean value used in requests where the Batch Count is greater than 1. If True, then batched
1597 operations SHALL be executed in the order in which they appear within the request. If False, then the
1598 server MAY choose to execute the batched operations in any order. If not specified, then False is
1599 assumed (i.e., no implied ordering). Server support for this feature is OPTIONAL, but if the server does
1600 not support the feature, and a request is received with the batch order option set to True, then the entire
1601 request SHALL be rejected.

Object	Encoding
Batch Order Option	Boolean

1602 **Table 180: Batch Order Option in Message Request Header**

1603 6.13 Batch Error Continuation Option

1604 This option SHALL only be present if the Batch Count is greater than 1. This option SHALL have one of
1605 three values:

- 1606 • *Undo* – If any operation in the request fails, then the server SHALL undo all the previous
1607 operations.
- 1608 • *Stop* – If an operation fails, then the server SHALL NOT continue processing subsequent
1609 operations in the request. Completed operations SHALL NOT be undone.
- 1610 • *Continue* – Return an error for the failed operation, and continue processing subsequent
1611 operations in the request.

1612 If not specified, then Stop is assumed.

1613 Server support for this feature is OPTIONAL, but if the server does not support the feature, and a request
1614 is received containing the *Batch Error Continuation Option* with a value other than the default Stop, then
1615 the entire request SHALL be rejected.

Object	Encoding
Batch Error Continuation	Enumeration, see 9.1.3.2.29

Option	
--------	--

1616 **Table 181: Batch Error Continuation Option in Message Request Header**

1617 6.14 Batch Count

1618 This field contains the number of Batch Items in a message and is REQUIRED. If only a single operation
 1619 is being requested, then the batch count SHALL be set to 1. The Message Payload, which follows the
 1620 Message Header, contains one or more batch items.

Object	Encoding
Batch Count	Integer

1621 **Table 182: Batch Count in Message Header**

1622 6.15 Batch Item

1623 This field consists of a structure that holds the individual requests or responses in a batch, and is
 1624 REQUIRED. The contents of the batch items are described in Section 7.2.

Object	Encoding
Batch Item	Structure

1625 **Table 183: Batch Item in Message**

1626 6.16 Message Extension

1627 The *Message Extension* is an OPTIONAL structure that MAY be appended to any Batch Item. It is used
 1628 to extend protocol messages for the purpose of adding vendor-specified extensions. The Message
 1629 Extension is a structure that SHALL contain the Vendor Identification, Criticality Indicator, and Vendor
 1630 Extension fields. The *Vendor Identification* SHALL be a text string that uniquely identifies the vendor,
 1631 allowing a client to determine if it is able to parse and understand the extension. If a client or server
 1632 receives a protocol message containing a message extension that it does not understand, then its actions
 1633 depend on the *Criticality Indicator*. If the indicator is True (i.e., Critical), and the receiver does not
 1634 understand the extension, then the receiver SHALL reject the entire message. If the indicator is False
 1635 (i.e., Non-Critical), and the receiver does not understand the extension, then the receiver MAY process
 1636 the rest of the message as if the extension were not present. The *Vendor Extension* structure SHALL
 1637 contain vendor-specific extensions.

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

1638 **Table 184: Message Extension Structure in Batch Item**

1639 **7 Message Format**

1640 Messages contain the following objects and fields. All fields SHALL appear in the order specified.

1641 **7.1 Message Structure**

Object	Encoding	REQUIRED
Request Message	Structure	
Request Header	Structure, see Table 187	Yes
Batch Item	Structure, see Table 188	Yes, MAY be repeated

1642 **Table 185: Request Message Structure**

Object	Encoding	REQUIRED
Response Message	Structure	
Response Header	Structure, see Table 189	Yes
Batch Item	Structure, see Table 190	Yes, MAY be repeated

1643 **Table 186: Response Message Structure**

1644 **7.2 Operations**

1645 If the client is capable of accepting asynchronous responses, then it MAY set the *Asynchronous Indicator*
 1646 in the header of a batched request. The batched responses MAY contain a mixture of synchronous and
 1647 asynchronous responses.

Request Header		
Object	REQUIRED in Message	Comment
Request Header	Yes	Structure
Protocol Version	Yes	See 6.1
Maximum Response Size	No	See 6.3
Asynchronous Indicator	No	If present, 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

1648 **Table 187: Request Header Structure**

Request Batch Item		
Object	REQUIRED in Message	Comment
Batch Item	Yes	Structure, see 6.15
Operation	Yes	See 6.2
Unique Batch Item ID	No	REQUIRED if <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

1649

Table 188: Request Batch Item Structure

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

1650

Table 189: Response Header Structure

Response Batch Item		
Object	REQUIRED in Message	Comment
Batch Item	Yes	Structure, see 6.15
Operation	Yes, if specified in Request Batch Item	See 6.2
Unique Batch Item ID	No	REQUIRED if present in Request Batch Item, see 6.4
Result Status	Yes	See 6.9
Result Reason	Yes, if Result Status is <i>Failure</i>	REQUIRED if Result Status is <i>Failure</i> , otherwise OPTIONAL, see 6.10
Result Message	No	OPTIONAL if Result Status is not <i>Pending</i> or <i>Success</i> , see 6.11
Asynchronous Correlation Value	No	REQUIRED 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

1651

Table 190: Response Batch Item Structure

1652

8 Authentication

1653

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

1654

1655

1656 9 Message Encoding

1657 To support different transport protocols and different client capabilities, a number of message-encoding
1658 mechanisms are supported.

1659 9.1 TTLV Encoding

1660 In order to minimize the resource impact on potentially low-function clients, one encoding mechanism to
1661 be used for protocol messages is a simplified TTLV (Tag, Type, Length, Value) scheme.

1662 The scheme is designed to minimize the CPU cycle and memory requirements of clients that need to
1663 encode or decode protocol messages, and to provide optimal alignment for both 32-bit and 64-bit
1664 processors. Minimizing bandwidth over the transport mechanism is considered to be of lesser importance.

1665 9.1.1 TTLV Encoding Fields

1666 Every Data object encoded by the TTLV scheme consists of four items, in order:

1667 9.1.1.1 Item Tag

1668 An Item Tag is a three-byte binary unsigned integer, transmitted big endian, which contains a number that
1669 designates the specific Protocol Field or Object that the TTLV object represents. To ease debugging, and
1670 to ensure that malformed messages are detected more easily, all tags SHALL contain either the value 42
1671 in hex or the value 54 in hex as the high order (first) byte. Tags defined by this specification contain hex
1672 42 in the first byte. Extensions, which are permitted, but are not defined in this specification, contain the
1673 value 54 hex in the first byte. A list of defined Item Tags is in Section 9.1.3.1

1674 9.1.1.2 Item Type

1675 An Item Type is a byte containing a coded value that indicates the data type of the data object. The
1676 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

1677 **Table 191: Allowed Item Type Values**

1678 **9.1.1.3 Item Length**

1679 An Item Length is a 32-bit binary integer, transmitted big-endian, containing the number of bytes in the
 1680 Item Value. The allowed values are:

1681

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 192: Allowed Item Length Values

1682

1683 If the Item Type is Structure, then the Item Length is the total length of all of the sub-items contained in
 1684 the structure, including any padding. If the Item Type is Integer, Enumeration, Text String, Byte String, or
 1685 Interval, then the Item Length is the number of bytes excluding the padding bytes. Text Strings and Byte
 1686 Strings SHALL be padded with the minimal number of bytes following the Item Value to obtain a multiple
 1687 of eight bytes. Integers, Enumerations, and Intervals SHALL be padded with four bytes following the Item
 1688 Value.

1689 **9.1.1.4 Item Value**

1690 The item value is a sequence of bytes containing the value of the data item, depending on the type:

- 1691 • Integers are encoded as four-byte long (32 bit) binary signed numbers in 2's complement
 1692 notation, transmitted big-endian.
- 1693 • Long Integers are encoded as eight-byte long (64 bit) binary signed numbers in 2's complement
 1694 notation, transmitted big-endian.
- 1695 • Big Integers are encoded as a sequence of eight-bit bytes, in two's complement notation,
 1696 transmitted big-endian. If the length of the sequence is not a multiple of eight bytes, then Big
 1697 Integers SHALL be padded with the minimal number of leading sign-extended bytes to make the
 1698 length a multiple of eight bytes. These padding bytes are part of the Item Value and SHALL be
 1699 counted in the Item Length.
- 1700 • Enumerations are encoded as four-byte long (32 bit) binary unsigned numbers transmitted big-
 1701 endian. Extensions, which are permitted, but are not defined in this specification, contain the
 1702 value 8 hex in the first nibble of the first byte.
- 1703 • Booleans are encoded as an eight-byte value that SHALL either contain the hex value
 1704 0000000000000000, indicating the Boolean value *False*, or the hex value 0000000000000001,
 1705 transmitted big-endian, indicating the Boolean value *True*.

- 1706 • Text Strings are sequences of bytes that encode character values according to the UTF-8
1707 encoding standard. There SHALL NOT be null-termination at the end of such strings.
- 1708 • Byte Strings are sequences of bytes containing individual unspecified eight-bit binary values, and
1709 are interpreted in the same sequence order.
- 1710 • Date-Time values are POSIX Time values encoded as Long Integers. POSIX Time, as described
1711 in IEEE Standard 1003.1 [IEEE1003-1], is the number of seconds since the Epoch (1970 Jan 1,
1712 00:00:00 UTC), not counting leap seconds.
- 1713 • Intervals are encoded as four-byte long (32 bit) binary unsigned numbers, transmitted big-endian.
1714 They have a resolution of one second.
- 1715 • Structure Values are encoded as the concatenated encodings of the elements of the structure. All
1716 structures defined in this specification SHALL have all of their fields encoded in the order in which
1717 they appear in their respective structure descriptions.

1718 9.1.2 Examples

1719 These examples are assumed to be encoding a Protocol Object whose tag is 420020. The examples are
1720 shown as a sequence of bytes in hexadecimal notation:

- 1721 • An Integer containing the decimal value 8:
1722 42 00 20 | 02 | 00 00 00 04 | 00 00 00 08 00 00 00 00
- 1723 • A Long Integer containing the decimal value 123456789000000000:
1724 42 00 20 | 03 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00
- 1725 • A Big Integer containing the decimal value 123456789000000000000000000000:
1726 42 00 20 | 04 | 00 00 00 10 | 00 00 00 00 03 FD 35 EB 6B C2 DF 46 18 08
1727 00 00
- 1728 • An Enumeration with value 255:
1729 42 00 20 | 05 | 00 00 00 04 | 00 00 00 FF 00 00 00 00
- 1730 • A Boolean with the value *True*:
1731 42 00 20 | 06 | 00 00 00 08 | 00 00 00 00 00 00 00 01
- 1732 • A Text String with the value "Hello World":
1733 42 00 20 | 07 | 00 00 00 0B | 48 65 6C 6C 6F 20 57 6F 72 6C 64 00 00 00
1734 00 00
- 1735 • A Byte String with the value { 0x01, 0x02, 0x03 }:
1736 42 00 20 | 08 | 00 00 00 03 | 01 02 03 00 00 00 00 00
- 1737 • A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:
1738 42 00 20 | 09 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8
- 1739 • An Interval, containing the value for 10 days:
1740 42 00 20 | 0A | 00 00 00 04 | 00 0D 2F 00 00 00 00 00
- 1741 • A Structure containing an Enumeration, value 254, followed by an Integer, value 255, having tags
1742 420004 and 420005 respectively:
1743 42 00 20 | 01 | 00 00 00 20 | 42 00 04 | 05 | 00 00 00 04 | 00 00 00 FE
1744 00 00 00 00 | 42 00 05 | 02 | 00 00 00 04 | 00 00 00 FF 00 00 00 00

1745 **9.1.3 Defined Values**

1746 This section specifies the values that are defined by this specification. In all cases where an extension
1747 mechanism is allowed, this extension mechanism is only able to be used for communication between
1748 parties that have pre-agreed understanding of the specific extensions.

1749 **9.1.3.1 Tags**

1750 The following table defines the tag values for the objects and primitive data values for the protocol
1751 messages.

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

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

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

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

Tag	
Object	Tag Value
Key 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 Count	420096
Usage Limits Total	420097
Usage Limits Unit	420098
Username	420099
Validity Date	42009A
Validity Indicator	42009B
Vendor Extension	42009C
Vendor Identification	42009D
Wrapping Method	42009E
X	42009F
Y	4200A0
Password	4200A1
(Reserved)	4200A2 - 42FFFF
(Unused)	430000 - 53FFFF
Extensions	540000 - 54FFFF
(Unused)	550000 - FFFFFF

Table 193: Tag Values

1753 **9.1.3.2 Enumerations**

1754 The following tables define the values for enumerated lists. Values not listed (outside the range 80000000
 1755 to 8FFFFFFF) are reserved for future KMIP versions.

1756 **9.1.3.2.1 Credential Type Enumeration**

Credential Type	
Name	Value
Username and Password	00000001
Extensions	8XXXXXXXX

1757 **Table 194: Credential Type Enumeration**

1758 **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

1759 **Table 195: Key Compression Type Enumeration**

1760 **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

1761

Table 196: Key Format Type Enumeration

1762

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

1763

Table 197: Wrapping Method Enumeration

1764

9.1.3.2.5 Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV

1765

Recommended curves are defined in **[FIPS186-3]**.

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

1766 **Table 198: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV**

1767 **9.1.3.2.6 Certificate Type Enumeration**

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

1768 **Table 199: Certificate Type Enumeration**

1769 **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

1770 **Table 200: Split Key Method Enumeration**

1771 **9.1.3.2.8 Secret Data Type Enumeration**

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

1772 **Table 201: Secret Data Type Enumeration**

1773 **9.1.3.2.9 Opaque Data Type Enumeration**

Opaque Data Type	
Name	Value
Extensions	8XXXXXXXX

1774 **Table 202: Opaque Data Type Enumeration**

1775 **9.1.3.2.10 Name Type Enumeration**

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

1776 **Table 203: Name Type Enumeration**

1777 **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

1778 **Table 204: Object Type Enumeration**

1779 **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
Blowfish	00000010
Camellia	00000011
CAST5	00000012
IDEA	00000013
MARS	00000014
RC2	00000015
RC4	00000016
RC5	00000017
SKIPJACK	00000018
Twofish	00000019
Extensions	8XXXXXXXX

Table 205: Cryptographic Algorithm Enumeration

1780

1781 **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

1782 **Table 206: Block Cipher Mode Enumeration**

1783 **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

1784 **Table 207: Padding Method Enumeration**

1785 **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
RIPEND-160	00000009
Tiger	0000000A
Whirlpool	0000000B
Extensions	8XXXXXXXX

1786 **Table 208: Hashing Algorithm Enumeration**

1787 **9.1.3.2.16 Key Role Type Enumeration**

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

Table 209: Key Role Type Enumeration

1788
 1789 Note that while the set and definitions of key role types are chosen to match TR-31 there is no necessity
 1790 to match binary representations.

1791 **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
------------	-----------

1792

Table 210: State Enumeration

1793

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

1794

Table 211: Revocation Reason Code Enumeration

1795

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

1796

Table 212: Link Type Enumeration

1797

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

1798 **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

1799 **Table 213: Derivation Method Enumeration**

1800 **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

1801 **Table 214: Certificate Request Type Enumeration**

1802 **9.1.3.2.22 Validity Indicator Enumeration**

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

1803 **Table 215: Validity Indicator Enumeration**

1804 **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

1805

Table 216: Query Function Enumeration

1806

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

1807

Table 217: Cancellation Result Enumeration

1808

9.1.3.2.25 Put Function Enumeration

Put Function	
Name	Value
New	00000001
Replace	00000002
Extensions	8XXXXXXXX

1809

Table 218: Put Function Enumeration

1810 **9.1.3.2.26 Operation Enumeration**

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

Table 219: Operation Enumeration

1811

1812 **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

1813 **Table 220: Result Status Enumeration**

1814 **9.1.3.2.28 Result Reason Enumeration**

Result Reason	
Name	Value
Item Not Found	00000001
Response Too Large	00000002
Authentication Not Successful	00000003
Invalid Message	00000004
Operation Not Supported	00000005
Missing Data	00000006
Invalid Field	00000007
Feature Not Supported	00000008
Operation Canceled By Requester	00000009
Cryptographic Failure	0000000A
Illegal Operation	0000000B
Permission Denied	0000000C
Object archived	0000000D
Index Out of Bounds	0000000E
Application Namespace Not Supported	0000000F
Key Format Type Not Supported	00000010
Key Compression Type Not Supported	00000011
General Failure	00000100
Extensions	8XXXXXXXX

1815 **Table 221: Result Reason Enumeration**

1816 **9.1.3.2.29 Batch Error Continuation Option Enumeration**

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

1817 **Table 222: Batch Error Continuation Option Enumeration**

1818 **9.1.3.2.30 Usage Limits Unit Enumeration**

Usage Limits Unit	
Name	Value
Byte	00000001
Object	00000002
Extensions	8XXXXXXXX

1819 **Table 223: Usage Limits Unit Enumeration**

1820

1821 **9.1.3.3 Bit Masks**

1822 **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

1823 **Table 224: Cryptographic Usage Mask**

1824 This list takes into consideration values which MAY appear in the Key Usage extension in an X.509
 1825 certificate.

1826 **9.1.3.3.2 Storage Status Mask**

Storage Status Mask	
Name	Value
On-line storage	00000001
Archival storage	00000002
Extensions	XXXXXXXX0

1827 **Table 225: Storage Status Mask**

1828 **9.2 XML Encoding**

1829 An XML Encoding has not yet been defined.

1830 **10 Transport**

1831 A KMIP Server SHALL establish and maintain channel confidentiality and integrity, and provide
1832 assurance of server authenticity for KMIP messaging.

1833 If a KMIP Server uses TCP/IP for KMIP messaging, then it SHALL support TLS v1.0 [RFC 2246] or later
1834 and may support other protocols as specified in [KMIP-Prof].

11 Error Handling

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

11.1 General

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

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

	Operation Failed	
--	------------------	--

1840

Table 226: General Errors

1841

11.2 Create

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

1842

Table 227: Create Errors

1843

11.3 Create Key Pair

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

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

1844

Table 228: Create Key Pair Errors

1845 **11.4 Register**

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

1846

Table 229: Register Errors

1847 **11.5 Re-key**

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object specified is not able to be re-keyed	Operation Failed	Permission Denied
Offset field is not permitted to be specified at the same time as any of the Activation Date, Process Start Date, Protect Stop Date, or Deactivation Date	Operation Failed	Invalid Message

attributes		
Cryptographic error during re-key	Operation Failed	Cryptographic Failure
The particular Application Namespace is not supported, and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived
An offset cannot be used to specify new Process Start, Protect Stop and/or Deactivation Date attribute values since no Activation Date has been specified for the existing key	Operation Failed	Illegal Operation

1848

Table 230: Re-key Errors

1849

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
The specified length exceeds the output of the derivation method or other cryptographic error during derivation.	Operation Failed	Cryptographic Failure

1850

Table 231: Derive Key Errors-1851 **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
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

1852

Table 232: Certify Errors1853 **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
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

1854

Table 233: Re-certify Errors1855 **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	Operation Failed	Invalid Field

the request		
-------------	--	--

1856

Table 234: Locate Errors

1857

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
Check cannot be performed on this object	Operation Failed	Illegal Operation
The client is not allowed to use the object according to the specified attributes	Operation Failed	Permission Denied

1858

Table 235: Check Errors

1859

11.11 Get

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

1860

Table 236: Get Errors

1861 **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

1862 **Table 237: Get Attributes Errors**

1863 **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

1864 **Table 238: Get Attribute List Errors**

1865 **11.14 Add Attribute**

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

1866 **Table 239: Add Attribute Errors**

1867

11.15 Modify Attribute

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

1868

Table 240: Modify Attribute Errors

1869

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
Attribute Index is not specified and the attribute has multiple instances	Operation Failed	Invalid Field

1870

Table 241: Delete Attribute Errors

1871 **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

1872 **Table 242: Obtain Lease Errors**

1873 **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
No Usage Limits Count is specified	Operation Failed	Invalid Message
Object is archived	Operation Failed	Object Archived
The server was not able to grant the requested amount of usage allocation	Operation Failed	Permission Denied

1874 **Table 243: Get Usage Allocation Errors**

1875 **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

1876 **Table 244: Activate Errors**

1877 **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

1878 **Table 245: Revoke Errors**

1879 **11.21 Destroy**

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

1880 **Table 246: Destroy Errors**

1881 **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

1882 **Table 247: Archive Errors**

1883 **11.23 Recover**

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

1884 **Table 248: Recover Errors**

1885 **11.24 Validate**

Error Definition	Result Status	Result Reason
The combination of Certificate Objects	Operation Failed	Invalid Message

and Unique Identifiers does not specify a certificate list		
One or more of the objects is archived	Operation Failed	Object Archived

1886

Table 249: Validate Errors

1887 **11.25 Query**

1888 N/A

1889 **11.26 Cancel**

1890 N/A

1891 **11.27 Poll**

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

1892

Table 250: Poll Errors

1893 **11.28 Batch Items**

1894 These errors MAY occur when a protocol message with one or more batch items is processed by the
 1895 server. If a message with one or more batch items was parsed correctly, then the response message
 1896 SHOULD include response(s) to the batch item(s) in the request according to the table below.

1897

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

1898

Table 251: Batch Items Errors

1899

12 Server Baseline Implementation Conformance Profile

1900

1901 The intention of the baseline conformance profile is for the minimal KMIP Server to support the
1902 mechanics of communication and to support a limited set of commands, such as query. The minimal
1903 KMIP Server would not need to support any particular algorithm – this would be the work of additional
1904 profiles.

1905 An implementation is a conforming KMIP Server if the implementation meets the conditions in Section
1906 12.1.

1907 An implementation SHALL be a conforming KMIP Server.

1908 If an implementation claims support for a particular clause, then the implementation SHALL conform to all
1909 normative statements within that clause and any subclauses to that clause.

12.1 Conformance clauses for a KMIP Server

1910

1911 An implementation conforms to this specification as a KMIP Server if it meets the following conditions:

1912

1. Supports the following objects:

1913

a. Attribute (see 2.1.1)

1914

b. Credential (see 2.1.2)

1915

c. Key Block (see 2.1.3)

1916

d. Key Value (see 2.1.4)

1917

e. Template-Attribute Structure (see 2.1.8)

1918

2. Supports the following attributes:

1919

a. Unique Identifier (see 3.1)

1920

b. Name (see 3.2)

1921

c. Object Type (see 3.3)

1922

d. Cryptographic Algorithm (see 3.4)

1923

e. Cryptographic Length (see 3.5)

1924

f. Cryptographic Parameters (see 3.6)

1925

g. Digest (see 3.12)

1926

h. Default Operation Policy (see 3.13.2)

1927

i. Cryptographic Usage Mask (see 3.14)

1928

j. State (see 3.17)

1929

k. Initial Date (see 3.18)

1930

l. Activation Date (see 3.19)

1931

m. Deactivation Date (see 3.22)

1932

n. Compromise Occurrence Date (see 3.24)

1933

o. Compromise Date (see 3.25)

1934

p. Revocation Reason (see 3.26)

1935

q. Last Change Date (see 3.32)

1936

3. Supports the ID Placeholder (see 4)

1937

4. Supports the following client-to-server operations:

1938

a. Locate (see 4.8)

- 1939 b. Check (see 4.9)
- 1940 c. Get (see 4.10)
- 1941 d. Get Attribute (see 4.11)
- 1942 e. Get Attribute List (see 4.12)
- 1943 f. Add Attribute (see 4.13)
- 1944 g. Modify Attribute (see 4.14)
- 1945 h. Delete Attribute (see 4.15)
- 1946 i. Activate (see 4.18)
- 1947 j. Revoke (see 4.19)
- 1948 k. Destroy (see 4.20)
- 1949 l. Query (see 4.24)
- 1950 5. Supports the following message contents:
 - 1951 a. Protocol Version (see 6.1)
 - 1952 b. Operation (see 6.2)
 - 1953 c. Maximum Response Size (see 6.3)
 - 1954 d. Unique Batch Item ID (see 6.4)
 - 1955 e. Time Stamp (see 6.5)
 - 1956 f. Asynchronous Indicator (see 6.7)
 - 1957 g. Result Status (see 6.9)
 - 1958 h. Result Reason (see 6.10)
 - 1959 i. Batch Order Option (see 6.12)
 - 1960 j. Batch Error Continuation Option (see 6.13)
 - 1961 k. Batch Count (see 6.14)
 - 1962 l. Batch Item (see 6.15)
- 1963 6. Supports Message Format (see 7)
- 1964 7. Supports Authentication (see 8)
- 1965 8. Supports the TTLV encoding (see 9.1)
- 1966 9. Supports the transport requirements (see 10)
- 1967 10. Supports Error Handling (see 11) for any supported object, attribute, or operation
- 1968 11. Optionally supports any clause within this specification that is not listed above
- 1969 12. Optionally supports extensions outside the scope of this standard (e.g., vendor extensions,
1970 conformance profiles) that do not contradict any requirements within this standard
- 1971 13. Supports at least one of the profiles defined in the KMIP Profiles Specification **[KMIP-Prof]**.
- 1972

1973

A. Attribute Cross-reference

1974

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

1975

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

1976

Table 252: Attribute Cross-reference

1977

B. Tag Cross-reference

1978

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	*	type varies
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 Role Type	3.6, 9.1.3.2.16	Enumeration	
Key Value	2.1.4	Byte String / Structure	
Key Wrapping Data	2.1.5	Structure	

Object	Defined	Type	Notes
Key Wrapping Specification	2.1.6	Structure	
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	
Password	2.1.2	Text String	
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	

Object	Defined	Type	Notes
Public Exponent	2.1.7	Big Integer	
Public Key	2.2.3	Structure	
Public Key Template-Attribute	2.1.8	Structure	
Public Key Unique Identifier	4.2	Text String	
Put Function	5.2, 9.1.3.2.25	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	Structure	
Request Message	7.1	Structure	
Request Payload	4, 5, 7.2	Structure	
Response Header	7.2	Structure	
Response Message	7.1	Structure	
Response Payload	4, 7.2	Structure	
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	
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	

Object	Defined	Type	Notes
Transparent*	2.1.7	Structure	
Unique Identifier	3.1	Text String	
Unique Batch Item ID	6.4	Byte String	
Username	2.1.2	Text String	
Usage Limits	3.16	Structure	
Usage Limits Count	3.16	Long Integer	
Usage Limits Total	3.16	Long Integer	
Usage Limits Unit	3.16	Enumeration	
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	

1979

Table 253: Tag Cross-reference

1980

1981

C. Operation and Object Cross-reference

1982

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

1983

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

1984

Table 254: Operation and Object Cross-reference

1985 **D. Acronyms**

1986 The following abbreviations and acronyms are used in this document:

1987	3DES	- Triple Data Encryption Standard specified in ANSI X9.52
1988	AES	- Advanced Encryption Standard specified in FIPS 197
1989	ASN.1	- Abstract Syntax Notation One specified in ITU-T X.680
1990	BDK	- Base Derivation Key specified in ANSI X9 TR-31
1991	CA	- Certification Authority
1992	CBC	- Cipher Block Chaining
1993	CCM	- Counter with CBC-MAC specified in NIST SP 800-38C
1994	CFB	- Cipher Feedback specified in NIST SP 800-38A
1995	CMAC	- Cipher-based MAC specified in NIST SP 800-38B
1996	CMC	- Certificate Management Messages over CMS specified in RFC 5275
1997	CMP	- Certificate Management Protocol specified in RFC 4210
1998	CPU	- Central Processing Unit
1999	CRL	- Certificate Revocation List specified in RFC 5280
2000	CRMF	- Certificate Request Message Format specified in RFC 4211
2001	CRT	- Chinese Remainder Theorem
2002	CTR	- Counter specified in NIST SP 800-38A
2003	CVK	- Card Verification Key specified in ANSI X9 TR-31
2004	DEK	- Data Encryption Key
2005	DER	- Distinguished Encoding Rules specified in ITU-T X.690
2006	DES	- Data Encryption Standard specified in FIPS 46-3
2007	DH	- Diffie-Hellman specified in ANSI X9.42
2008	DNS	- Domain Name Server
2009	DSA	- Digital Signature Algorithm specified in FIPS 186-3
2010	DSKPP	- Dynamic Symmetric Key Provisioning Protocol
2011	ECB	- Electronic Code Book
2012	ECDH	- Elliptic Curve Diffie-Hellman specified in ANSI X9.63 and NIST SP 800-56A
2013	ECDSA	- Elliptic Curve Digital Signature Algorithm specified in ANSX9.62
2014	ECMQV	- Elliptic Curve Menezes Qu Vanstone specified in ANSI X9.63 and NIST SP 800-56A
2015	FFC	- Finite Field Cryptography
2016	FIPS	- Federal Information Processing Standard
2017	GCM	- Galois/Counter Mode specified in NIST SP 800-38D
2018	GF	- Galois field (or finite field)
2019	HMAC	- Keyed-Hash Message Authentication Code specified in FIPS 198-1 and RFC 2104
2020	HTTP	- Hyper Text Transfer Protocol
2021	HTTP(S)	- Hyper Text Transfer Protocol (Secure socket)

2022	IEEE	- Institute of Electrical and Electronics Engineers
2023	IETF	- Internet Engineering Task Force
2024	IP	- Internet Protocol
2025	IPsec	- Internet Protocol Security
2026	IV	- Initialization Vector
2027	KEK	- Key Encryption Key
2028	KMIP	- Key Management Interoperability Protocol
2029	MAC	- Message Authentication Code
2030	MKAC	- EMV/chip card Master Key: Application Cryptograms specified in ANSI X9 TR-31
2031	MKCP	- EMV/chip card Master Key: Card Personalization specified in ANSI X9 TR-31
2032	MKDAC	- EMV/chip card Master Key: Data Authentication Code specified in ANSI X9 TR-31
2033	MKDN	- EMV/chip card Master Key: Dynamic Numbers specified in ANSI X9 TR-31
2034	MKOTH	- EMV/chip card Master Key: Other specified in ANSI X9 TR-31
2035	MKSMC	- EMV/chip card Master Key: Secure Messaging for Confidentiality specified in X9 TR-31
2036	MKSMI	- EMV/chip card Master Key: Secure Messaging for Integrity specified in ANSI X9 TR-31
2037	MD2	- Message Digest 2 Algorithm specified in RFC 1319
2038	MD4	- Message Digest 4 Algorithm specified in RFC 1320
2039	MD5	- Message Digest 5 Algorithm specified in RFC 1321
2040	NIST	- National Institute of Standards and Technology
2041	OAEP	- Optimal Asymmetric Encryption Padding specified in PKCS#1
2042	OFB	- Output Feedback specified in NIST SP 800-38A
2043	PBKDF2	- Password-Based Key Derivation Function 2 specified in RFC 2898
2044	PCBC	- Propagating Cipher Block Chaining
2045	PEM	- Privacy Enhanced Mail specified in RFC 1421
2046	PGP	- OpenPGP specified in RFC 4880
2047	PKCS	- Public-Key Cryptography Standards
2048	PKCS#1	- RSA Cryptography Specification Version 2.1 specified in RFC 3447
2049	PKCS#5	- Password-Based Cryptography Specification Version 2 specified in RFC 2898
2050	PKCS#8	- Private-Key Information Syntax Specification Version 1.2 specified in RFC 5208
2051	PKCS#10	- Certification Request Syntax Specification Version 1.7 specified in RFC 2986
2052	POSIX	- Portable Operating System Interface
2053	RFC	- Request for Comments documents of IETF
2054	RSA	- Rivest, Shamir, Adelman (an algorithm)
2055	SCEP	- Simple Certificate Enrollment Protocol
2056	SCVP	- Server-based Certificate Validation Protocol
2057	SHA	- Secure Hash Algorithm specified in FIPS 180-2
2058	SP	- Special Publication
2059	SSL/TLS	- Secure Sockets Layer/Transport Layer Security

2060	S/MIME	- Secure/Multipurpose Internet Mail Extensions
2061	TDEA	- see 3DES
2062	TCP	- Transport Control Protocol
2063	TTLV	- Tag, Type, Length, Value
2064	URI	- Uniform Resource Identifier
2065	UTC	- Coordinated Universal Time
2066	UTF-8	- Universal Transformation Format 8-bit specified in RFC 3629
2067	XKMS	- XML Key Management Specification
2068	XML	- Extensible Markup Language
2069	XTS	- XEX Tweakable Block Cipher with Ciphertext Stealing specified in NIST SP 800-38E
2070	X.509	- Public Key Certificate specified in RFC 5280
2071	ZPK	- PIN Block Encryption Key specified in ANSI X9 TR-31

2072

E. List of Figures and Tables

2073	Figure 1: Cryptographic Object States and Transitions.....	46
2074		
2075	Table 1: Terminology	11
2076	Table 2: Attribute Object Structure	15
2077	Table 3: Credential Object Structure.....	16
2078	Table 4: Credential Value Structure for the Username and Password Credential.....	16
2079	Table 5: Key Block Object Structure	17
2080	Table 6: Key Value Object Structure.....	18
2081	Table 7: Key Wrapping Data Object Structure	19
2082	Table 8: Encryption Key Information Object Structure	19
2083	Table 9: MAC/Signature Key Information Object Structure.....	19
2084	Table 10: Key Wrapping Specification Object Structure	20
2085	Table 11: Parameter mapping.....	21
2086	Table 12: Key Material Object Structure for Transparent Symmetric Keys	21
2087	Table 13: Key Material Object Structure for Transparent DSA Private Keys	22
2088	Table 14: Key Material Object Structure for Transparent DSA Public Keys	22
2089	Table 15: Key Material Object Structure for Transparent RSA Private Keys	22
2090	Table 16: Key Material Object Structure for Transparent RSA Public Keys	23
2091	Table 17: Key Material Object Structure for Transparent DH Private Keys.....	23
2092	Table 18: Key Material Object Structure for Transparent DH Public Keys	23
2093	Table 19: Key Material Object Structure for Transparent ECDSA Private Keys.....	24
2094	Table 20: Key Material Object Structure for Transparent ECDSA Public Keys	24
2095	Table 21: Key Material Object Structure for Transparent ECDH Private Keys.....	24
2096	Table 22: Key Material Object Structure for Transparent ECDH Public Keys	24
2097	Table 23: Key Material Object Structure for Transparent ECMQV Private Keys.....	25
2098	Table 24: Key Material Object Structure for Transparent ECMQV Public Keys	25
2099	Table 25: Template-Attribute Object Structure.....	25
2100	Table 26: Certificate Object Structure	26
2101	Table 27: Symmetric Key Object Structure	26
2102	Table 28: Public Key Object Structure	26
2103	Table 29: Private Key Object Structure.....	26
2104	Table 30: Split Key Object Structure	27
2105	Table 31: Template Object Structure	28
2106	Table 32: Secret Data Object Structure	29
2107	Table 33: Opaque Object Structure	29
2108	Table 34: Attribute Rules.....	31
2109	Table 35: Unique Identifier Attribute.....	31
2110	Table 36: Unique Identifier Attribute Rules.....	32
2111	Table 37: Name Attribute Structure.....	32

2112	Table 38: Name Attribute Rules	32
2113	Table 39: Object Type Attribute	33
2114	Table 40: Object Type Attribute Rules	33
2115	Table 41: Cryptographic Algorithm Attribute	33
2116	Table 42: Cryptographic Algorithm Attribute Rules	33
2117	Table 43: Cryptographic Length Attribute	33
2118	Table 44: Cryptographic Length Attribute Rules	34
2119	Table 45: Cryptographic Parameters Attribute Structure	34
2120	Table 46: Cryptographic Parameters Attribute Rules	34
2121	Table 47: Key Role Types.....	35
2122	Table 48: Cryptographic Domain Parameters Attribute Structure	36
2123	Table 49: Cryptographic Domain Parameters Attribute Rules.....	36
2124	Table 50: Certificate Type Attribute.....	36
2125	Table 51: Certificate Type Attribute Rules.....	36
2126	Table 52: Certificate Identifier Attribute Structure	37
2127	Table 53: Certificate Identifier Attribute Rules.....	37
2128	Table 54: Certificate Subject Attribute Structure	37
2129	Table 55: Certificate Subject Attribute Rules.....	38
2130	Table 56: Certificate Issuer Attribute Structure	38
2131	Table 57: Certificate Issuer Attribute Rules.....	38
2132	Table 58: Digest Attribute Structure	39
2133	Table 59: Digest Attribute Rules	39
2134	Table 60: Operation Policy Name Attribute	39
2135	Table 61: Operation Policy Name Attribute Rules.....	40
2136	Table 62: Default Operation Policy for Secret Objects	41
2137	Table 63: Default Operation Policy for Certificates and Public Key Objects.....	42
2138	Table 64: Default Operation Policy for Private Template Objects	42
2139	Table 65: Default Operation Policy for Public Template Objects.....	43
2140	Table 66: X.509 Key Usage to Cryptographic Usage Mask Mapping	44
2141	Table 67: Cryptographic Usage Mask Attribute.....	44
2142	Table 68: Cryptographic Usage Mask Attribute Rules	44
2143	Table 69: Lease Time Attribute	44
2144	Table 70: Lease Time Attribute Rules.....	45
2145	Table 71: Usage Limits Attribute Structure.....	45
2146	Table 72: Usage Limits Attribute Rules.....	46
2147	Table 73: State Attribute	47
2148	Table 74: State Attribute Rules	48
2149	Table 75: Initial Date Attribute.....	48
2150	Table 76: Initial Date Attribute Rules.....	48
2151	Table 77: Activation Date Attribute.....	49
2152	Table 78: Activation Date Attribute Rules.....	49
2153	Table 79: Process Start Date Attribute.....	49

2154	Table 80: Process Start Date Attribute Rules.....	50
2155	Table 81: Protect Stop Date Attribute	50
2156	Table 82: Protect Stop Date Attribute Rules	51
2157	Table 83: Deactivation Date Attribute	51
2158	Table 84: Deactivation Date Attribute Rules	51
2159	Table 85: Destroy Date Attribute.....	51
2160	Table 86: Destroy Date Attribute Rules.....	52
2161	Table 87: Compromise Occurrence Date Attribute	52
2162	Table 88: Compromise Occurrence Date Attribute Rules	52
2163	Table 89: Compromise Date Attribute.....	52
2164	Table 90: Compromise Date Attribute Rules.....	53
2165	Table 91: Revocation Reason Attribute Structure	53
2166	Table 92: Revocation Reason Attribute Rules	53
2167	Table 93: Archive Date Attribute	54
2168	Table 94: Archive Date Attribute Rules	54
2169	Table 95: Object Group Attribute	54
2170	Table 96: Object Group Attribute Rules	54
2171	Table 97: Link Attribute Structure.....	55
2172	Table 98: Link Attribute Structure Rules	55
2173	Table 99: Application Specific Information Attribute.....	56
2174	Table 100: Application Specific Information Attribute Rules.....	56
2175	Table 101: Contact Information Attribute	56
2176	Table 102: Contact Information Attribute Rules	57
2177	Table 103: Last Change Date Attribute.....	57
2178	Table 104: Last Change Date Attribute Rules.....	57
2179	Table 105 Custom Attribute	58
2180	Table 106: Custom Attribute Rules	58
2181	Table 107: Create Request Payload	60
2182	Table 108: Create Response Payload	60
2183	Table 109: Create Attribute Requirements.....	60
2184	Table 110: Create Key Pair Request Payload.....	61
2185	Table 111: Create Key Pair Response Payload.....	61
2186	Table 112: Create Key Pair Attribute Requirements	62
2187	Table 113: Register Request Payload	62
2188	Table 114: Register Response Payload.....	63
2189	Table 115: Register Attribute Requirements	63
2190	Table 116: Computing New Dates from Offset during Re-key	64
2191	Table 117: Re-key Attribute Requirements	64
2192	Table 118: Re-key Request Payload	65
2193	Table 119: Re-key Response Payload.....	65
2194	Table 120: Derive Key Request Payload	66
2195	Table 121: Derive Key Response Payload.....	67

2196	Table 122: Derivation Parameters Structure (Except PBKDF2)	67
2197	Table 123: PBKDF2 Derivation Parameters Structure	68
2198	Table 124: Certify Request Payload	68
2199	Table 125: Certify Response Payload.....	69
2200	Table 126: Computing New Dates from Offset during Re-certify	69
2201	Table 127: Re-certify Attribute Requirements	70
2202	Table 128: Re-certify Request Payload	70
2203	Table 129: Re-certify Response Payload.....	71
2204	Table 130: Locate Request Payload	72
2205	Table 131: Locate Response Payload	72
2206	Table 132: Check Request Payload.....	73
2207	Table 133: Check Response Payload.....	73
2208	Table 134: Get Request Payload	74
2209	Table 135: Get Response Payload	74
2210	Table 136: Get Attributes Request Payload	75
2211	Table 137: Get Attributes Response Payload	75
2212	Table 138: Get Attribute List Request Payload	75
2213	Table 139: Get Attribute List Response Payload	75
2214	Table 140: Add Attribute Request Payload	76
2215	Table 141: Add Attribute Response Payload	76
2216	Table 142: Modify Attribute Request Payload	76
2217	Table 143: Modify Attribute Response Payload	76
2218	Table 144: Delete Attribute Request Payload	77
2219	Table 145: Delete Attribute Response Payload	77
2220	Table 146: Obtain Lease Request Payload.....	77
2221	Table 147: Obtain Lease Response Payload.....	78
2222	Table 148: Get Usage Allocation Request Payload	78
2223	Table 149: Get Usage Allocation Response Payload	78
2224	Table 150: Activate Request Payload	79
2225	Table 151: Activate Response Payload	79
2226	Table 152: Revoke Request Payload.....	79
2227	Table 153: Revoke Response Payload.....	79
2228	Table 154: Destroy Request Payload	80
2229	Table 155: Destroy Response Payload.....	80
2230	Table 156: Archive Request Payload.....	80
2231	Table 157: Archive Response Payload	80
2232	Table 158: Recover Request Payload	81
2233	Table 159: Recover Response Payload.....	81
2234	Table 160: Validate Request Payload.....	81
2235	Table 161: Validate Response Payload	81
2236	Table 162: Query Request Payload	82
2237	Table 163: Query Response Payload	82

2238	Table 164: Cancel Request Payload.....	83
2239	Table 165: Cancel Response Payload.....	83
2240	Table 166: Poll Request Payload.....	83
2241	Table 167: Notify Message Payload	84
2242	Table 168: Put Message Payload	85
2243	Table 169: Protocol Version Structure in Message Header	86
2244	Table 170: Operation in Batch Item	86
2245	Table 171: Maximum Response Size in Message Request Header	86
2246	Table 172: Unique Batch Item ID in Batch Item	87
2247	Table 173: Time Stamp in Message Header.....	87
2248	Table 174: Authentication Structure in Message Header.....	87
2249	Table 175: Asynchronous Indicator in Message Request Header	87
2250	Table 176: Asynchronous Correlation Value in Response Batch Item.....	87
2251	Table 177: Result Status in Response Batch Item.....	88
2252	Table 178: Result Reason in Response Batch Item.....	89
2253	Table 179: Result Message in Response Batch Item	89
2254	Table 180: Batch Order Option in Message Request Header	89
2255	Table 181: Batch Error Continuation Option in Message Request Header	90
2256	Table 182: Batch Count in Message Header	90
2257	Table 183: Batch Item in Message.....	90
2258	Table 184: Message Extension Structure in Batch Item	90
2259	Table 185: Request Message Structure.....	91
2260	Table 186: Response Message Structure.....	91
2261	Table 187: Request Header Structure	91
2262	Table 188: Request Batch Item Structure	92
2263	Table 189: Response Header Structure.....	92
2264	Table 190: Response Batch Item Structure	92
2265	Table 191: Allowed Item Type Values.....	94
2266	Table 192: Allowed Item Length Values.....	95
2267	Table 193: Tag Values	101
2268	Table 194: Credential Type Enumeration	102
2269	Table 195: Key Compression Type Enumeration.....	102
2270	Table 196: Key Format Type Enumeration	103
2271	Table 197: Wrapping Method Enumeration.....	103
2272	Table 198: Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV	104
2273	Table 199: Certificate Type Enumeration.....	104
2274	Table 200: Split Key Method Enumeration.....	104
2275	Table 201: Secret Data Type Enumeration.....	105
2276	Table 202: Opaque Data Type Enumeration.....	105
2277	Table 203: Name Type Enumeration	105
2278	Table 204: Object Type Enumeration	105
2279	Table 205: Cryptographic Algorithm Enumeration	106

2280	Table 206: Block Cipher Mode Enumeration.....	107
2281	Table 207: Padding Method Enumeration.....	107
2282	Table 208: Hashing Algorithm Enumeration.....	108
2283	Table 209: Key Role Type Enumeration	109
2284	Table 210: State Enumeration	110
2285	Table 211: Revocation Reason Code Enumeration	110
2286	Table 212: Link Type Enumeration	110
2287	Table 213: Derivation Method Enumeration.....	111
2288	Table 214: Certificate Request Type Enumeration	111
2289	Table 215: Validity Indicator Enumeration.....	111
2290	Table 216: Query Function Enumeration	112
2291	Table 217: Cancellation Result Enumeration.....	112
2292	Table 218: Put Function Enumeration.....	112
2293	Table 219: Operation Enumeration	113
2294	Table 220: Result Status Enumeration	114
2295	Table 221: Result Reason Enumeration	114
2296	Table 222: Batch Error Continuation Option Enumeration	115
2297	Table 223: Usage Limits Unit Enumeration.....	115
2298	Table 224: Cryptographic Usage Mask.....	116
2299	Table 225: Storage Status Mask.....	116
2300	Table 226: General Errors.....	120
2301	Table 227: Create Errors.....	120
2302	Table 228: Create Key Pair Errors	121
2303	Table 229: Register Errors.....	121
2304	Table 230: Re-key Errors	122
2305	Table 231: Derive Key Errors-.....	123
2306	Table 232: Certify Errors.....	123
2307	Table 233: Re-certify Errors.....	123
2308	Table 234: Locate Errors.....	124
2309	Table 235: Check Errors	124
2310	Table 236: Get Errors	124
2311	Table 237: Get Attributes Errors	125
2312	Table 238: Get Attribute List Errors.....	125
2313	Table 239: Add Attribute Errors.....	125
2314	Table 240: Modify Attribute Errors	126
2315	Table 241: Delete Attribute Errors.....	126
2316	Table 242: Obtain Lease Errors	127
2317	Table 243: Get Usage Allocation Errors.....	127
2318	Table 244: Activate Errors.....	127
2319	Table 245: Revoke Errors	128
2320	Table 246: Destroy Errors.....	128
2321	Table 247: Archive Errors	128

2322	Table 248: Recover Errors	128
2323	Table 249: Validate Errors	129
2324	Table 250: Poll Errors	129
2325	Table 251: Batch Items Errors	129
2326	Table 252: Attribute Cross-reference	133
2327	Table 253: Tag Cross-reference	138
2328	Table 254: Operation and Object Cross-reference	139
2329		

2330

F. Acknowledgements

2331 The following individuals have participated in the creation of this specification and are gratefully
2332 acknowledged:

2333 **Original Authors of the initial contribution:**

2334 David Babcock, HP
2335 Steven Bade, IBM
2336 Paolo Bezoari, NetApp
2337 Mathias Björkqvist, IBM
2338 Bruce Brinson, EMC
2339 Christian Cachin, IBM
2340 Tony Crossman, Thales/nCipher
2341 Stan Feather, HP
2342 Indra Fitzgerald, HP
2343 Judy Furlong, EMC
2344 Jon Geater, Thales/nCipher
2345 Bob Griffin, EMC
2346 Robert Haas, IBM (editor)
2347 Timothy Hahn, IBM
2348 Jack Harwood, EMC
2349 Walt Hubis, LSI
2350 Glen Jaquette, IBM
2351 Jeff Kravitz, IBM (editor emeritus)
2352 Michael McIntosh, IBM
2353 Brian Metzger, HP
2354 Anthony Nadalin, IBM
2355 Elaine Palmer, IBM
2356 Joe Pato, HP
2357 René Pawlitzek, IBM
2358 Subhash Sankuratripati, NetApp
2359 Mark Schiller, HP
2360 Martin Skagen, Brocade
2361 Marcus Streets, Thales/nCipher
2362 John Tattan, EMC
2363 Karla Thomas, Brocade
2364 Marko Vukolić, IBM
2365 Steve Wierenga, HP

2366 **Participants:**

2367
2368 Mike Allen, PGP Corporation
2369 Gordon Arnold, IBM
2370 Todd Arnold, IBM
2371 Matthew Ball, Oracle Corporation
2372 Elaine Barker, NIST
2373 Peter Bartok, Venafi, Inc.
2374 Mathias Björkqvist, IBM
2375 Kevin Bocek, Thales e-Security
2376 Kelley Burgin, National Security Agency
2377 Jon Callas, PGP Corporation
2378 Tom Clifford, Symantec Corp.
2379 Graydon Dodson, Lexmark International Inc.
2380 Chris Dunn, SafeNet, Inc.
2381 Paul Earsy, SafeNet, Inc.
2382 Stan Feather, Hewlett-Packard

2383 Indra Fitzgerald, Hewlett-Packard
2384 Alan Frindell, SafeNet, Inc.
2385 Judith Furlong, EMC Corporation
2386 Jonathan Geater, Thales e-Security
2387 Robert Griffin, EMC Corporation
2388 Robert Haas, IBM
2389 Thomas Hardjono, M.I.T.
2390 Kurt Heberlein, 3PAR, Inc.
2391 Marc Hocking, BeCrypt Ltd.
2392 Larry Hofer, Emulex Corporation
2393 Brandon Hoff, Emulex Corporation
2394 Walt Hubis, LSI Corporation
2395 Tim Hudson, Cryptsoft
2396 Wyllys Ingersoll, Oracle Corporation
2397 Jay Jacobs, Target Corporation
2398 Glen Jaquette, IBM
2399 Scott Kipp, Brocade Communications Systems, Inc.
2400 David Lawson, Emulex Corporation
2401 Hal Lockhart, Oracle Corporation
2402 Robert Lockhart, Thales e-Security
2403 Shyam Mankala, EMC Corporation
2404 Upendra Mardikar, PayPal Inc.
2405 Marc Massar, Individual
2406 Don McAlister, Associate
2407 Hyrum Mills, Mitre Corporation
2408 Bob Nixon, Emulex Corporation
2409 Landon Curt Noll, Cisco Systems, Inc.
2410 René Pawlitzek, IBM
2411 John Peck, IBM
2412 Rob Philpott, EMC Corporation
2413 Scott Rea, Individual
2414 Bruce Rich, IBM
2415 Scott Rotondo, Oracle Corporation
2416 Saikat Saha, Vormetric, Inc.
2417 Anil Saldhana, Red Hat
2418 Subhash Sankuratiripati, NetApp
2419 Mark Schiller, Hewlett-Packard
2420 Jitendra Singh, Brocade Communications Systems, Inc.
2421 Servesh Singh, EMC Corporation
2422 Terence Spies, Voltage Security
2423 Sandy Stewart, Oracle Corporation
2424 Marcus Streets, Thales e-Security
2425 Brett Thompson, SafeNet, Inc.
2426 Benjamin Tomhave, Individual
2427 Sean Turner, IECA, Inc.
2428 Paul Turner, Venafi, Inc.
2429 Marko Vukolić, IBM
2430 Rod Wideman, Quantum Corporation
2431 Steven Wierenga, Hewlett-Packard
2432 Peter Yee, EMC Corporation
2433 Krishna Yellepeddy, IBM
2434 Peter Zelechowski, Election Systems & Software
2435 Grace Zhang, Skyworth TTG Holdings Limited
2436

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). Version used for Public Review 01.
draft-07	2010-02-04	Robert Haas	Editorial fixes according to Elaine Barker's comments. Comments for which the proposed resolution is "No Change" are indicated accordingly. Open issues marked with "TBD" and possible Usage Guide items are marked with "UG".
draft-08	2010-03-02	Robert Haas, Indra Fitzgerald	Incorporated TC and non-TC editorial and technical comments from the public review: Simplified Usage Limits attribute, added Template as a third parameter to Register, restricted custom attributes to have at most one level of structures (Matt Ball). Incorporated ballot changes towards server-to-server support, extended Get Attributes to allow returning all attributes, clarified Operation Policy Name attribute (Marko Vukolic). Clarified Transparent Key Structures (Judy Furlong). Clarified Cryptographic Domain Parameters and Create Key Pair (Elaine Barker).
draft-09	2010-03-15	Robert Haas, Indra Fitzgerald	Revised Credential object to specify Username and Password (Matt Ball). Clarified Transparent Key section with new parameter-mapping table (Indra Fitzgerald). Clarified Digest attribute description. Renamed Role Type to Key Role Type. Editorial fixes.
draft-10	2010-03-18	Robert Haas	Updated participants' list. Editorial fixes. Version used for Public Review 02.

draft-11	2010-05-25	Robert Haas, Indra Fitzgerald	Incorporated TC and non-TC editorial and technical comments from the second public review (from Tim Hudson, Bruce Rich, Mathias Bjoerkqvist, Elaine Barker, and Tony Stieber). Added details for PGP certificates in the Certificate object, and in the Certificate Subject and Certificate Identifier attributes. Added enumerations for the Cryptographic Algorithms and Hashing Algorithms. Editorial fixes.
draft-12		Robert Haas	Updated participants' list. Changed SHALL to SHOULD for distinct values in multi-instance attributes. Updated cross-references to KMIP specs. Version used for Committee Specification.

2438