

TBD (Key Management Interoperability Protocol)

Editor's Draft 0.98

17 September 2009

Deleted: 3

Specification URIs:

This Version:

[TBD.html](#)
[TBD.doc](#) (Authoritative)
[TBD.pdf](#)

Previous Version:

[TBD.html](#)
[TBD.doc](#) (Authoritative)
[TBD.pdf](#)

Latest Version:

[TBD.html](#)
[TBD.doc](#)
[TBD.pdf](#)

Technical Committee:

OASIS Key Management Interoperability Protocol (KMIP) TC

Chair(s):

Robert Griffin
Subhash Sankuratripati

Editor(s):

Robert Haas
Indra Fitzgerald

Related work:

This specification replaces or supersedes:

- None

This specification is related to:

- TBD

Declared XML Namespace(s):

TBD

Abstract:

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

Status:

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

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

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

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

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

Notices

Copyright © OASIS® 2009. All Rights Reserved.

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

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

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

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

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

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

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

The names "OASIS", [insert specific trademarked names and abbreviations here] 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 Document Roadmap	8
1.2 Goals and Requirements.....	8
1.3 Notational Conventions	8
1.4 Namespaces	8
1.5 Terminology	8
1.6 Normative References.....	9
1.7 Non-normative References	9
1.8 Compliance	9
2 Objects	10
2.1 Base Objects.....	10
2.1.1 Attribute	10
2.1.2 Credential	10
2.1.3 Key Block.....	11
2.1.4 Key Value	12
2.1.5 Key Wrapping Data.....	13
2.1.6 Key Wrapping Specification	14
2.1.7 Transparent Key Structures	15
2.1.8 Template-Attribute Structures	18
2.2 Managed Objects	18
2.2.1 Certificate.....	18
2.2.2 Symmetric Key.....	19
2.2.3 Public Key.....	19
2.2.4 Private Key	19
2.2.5 Split Key.....	19
2.2.6 Template.....	21
2.2.7 Secret Data.....	21
2.2.8 Opaque Object.....	22
3 Attributes	23
3.1 Unique Identifier	23
3.2 Name	23
3.3 Object Type.....	24
3.4 Cryptographic Algorithm.....	24
3.5 Cryptographic Length.....	25
3.6 Cryptographic Parameters	25
3.7 Certificate Type.....	27
3.8 Certificate Issuer	28
3.9 Certificate Subject.....	28
3.10 Digest.....	29
3.11 Operation Policy Name	30
3.11.1 Operations outside of operation policy control	30
3.11.2 Default Operation Policy	30
3.12 Cryptographic Usage Mask	33

3.13 Lease Time	34
3.14 Usage Limits	35
3.15 State	36
3.16 Initial Date	38
3.17 Activation Date	38
3.18 Process Start Date	39
3.19 Protect Stop Date	39
3.20 Deactivation Date	40
3.21 Destroy Date	41
3.22 Compromise Occurrence Date	41
3.23 Compromise Date	41
3.24 Revocation Reason	42
3.25 Archive Date	43
3.26 Object Group	43
3.27 Link	43
3.28 Application Specific Information	45
3.29 Contact Information	45
3.30 Last Changed Date	46
3.31 Custom Attribute	46
4 Client-to-Server Operations	47
4.1 Create	48
4.2 Create Key Pair	49
4.3 Register	50
4.4 Re-key	51
4.5 Derive Key	53
4.6 Certify	56
4.7 Re-certify	57
4.8 Locate	59
4.9 Check	60
4.10 Get	62
4.11 Get Attributes	62
4.12 Get Attribute List	63
4.13 Add Attribute	63
4.14 Modify Attribute	64
4.15 Delete Attribute	64
4.16 Obtain Lease	65
4.17 Get Usage Allocation	66
4.18 Activate	67
4.19 Revoke	67
4.20 Destroy	68
4.21 Archive	68
4.22 Recover	68
4.23 Validate	69
4.24 Query	69
4.25 Cancel	71

4.26 Poll	72
5 Server-to-Client Operations	72
5.1 Notify	72
5.2 Put	72
6 Message Contents	73
6.1 Protocol Version	73
6.2 Operation	74
6.3 Maximum Response Size	74
6.4 Unique Batch Item ID	74
6.5 Time Stamp	74
6.6 Authentication	75
6.7 Asynchronous Indicator	75
6.8 Asynchronous Correlation Value	75
6.9 Result Status	75
6.10 Result Reason	76
6.11 Result Message	76
6.12 Batch Order Option	77
6.13 Batch Error Continuation Option	77
6.14 Batch Count	77
6.15 Batch Item	77
6.16 Message Extension	78
7 Message Format	78
7.1 Message Structure	78
7.2 Synchronous Operations	79
7.3 Asynchronous Operations	80
8 Authentication	81
9 Message Encoding	82
9.1 TTLV Encoding	82
9.1.1 TTLV Encoding Fields	82
9.1.2 Examples	84
9.1.3 Defined Values	85
9.2 XML Encoding	104
10 Transport	104
10.1 SSL/TLS Profile	104
10.1.1 Mandatory cipher suites	104
10.1.2 Discouraged cipher suites	104
10.2 HTTPS Profile	105
10.2.1 HTTP Encoding	105
11 Error Handling	105
11.1 General	106
11.2 Create	106
11.3 Create Key Pair	107
11.4 Register	108
11.5 Re-key	108
11.6 Derive Key	109

11.7 Certify.....	109
11.8 Re-certify.....	110
11.9 Locate	110
11.10 Check.....	110
11.11 Get	111
11.12 Get Attributes	111
11.13 Get Attribute List	111
11.14 Add Attribute	112
11.15 Modify Attribute	112
11.16 Delete Attribute	113
11.17 Obtain Lease.....	113
11.18 Get Usage Allocation.....	113
11.19 Activate	114
11.20 Revoke.....	114
11.21 Destroy.....	114
11.22 Archive	114
11.23 Recover.....	115
11.24 Validate	115
11.25 Query	115
11.26 Cancel.....	115
11.27 Poll.....	115
11.28 Batch Items	115
12 Security Considerations.....	116
A. Attribute Cross-reference	117
B. Tag Cross-reference	119
C. Operation and Object Cross-reference	124
D. Acronyms.....	125
E. List of Figures and Tables.....	127
F. Acknowledgements	134
G. Revision History.....	135

1 Introduction

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

The protocol specified in this document includes several certificate-related functions for which there are a number of existing protocols – namely Validate (e.g., SVP or XKMS), Certify (e.g. CMP, CMC, SCEP) and Re-certify (e.g. CMP, CMC, SCEP). The protocol does not attempt to define a comprehensive certificate management protocol such as would be **REQUIRED** 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
- Authentication profiles for clients and servers
- Message encoding (including enumerations)
- Error handling

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

1.1 Document Roadmap

TBD

1.2 Goals and Requirements

TBD

1.3 Notational Conventions

TBD

1.4 Namespaces

TBD

1.5 Terminology

TBD

Deleted: (i.e.,

Deleted: required

Deleted:)

37 **1.6 Normative References**

38 TBD

39 RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997.

40 **1.7 Non-normative References**

41 TBD

42 **1.8 Compliance**

43 TBD

44 The key words "SHALL", "SHALL NOT", "REQUIRED", "SHOULD", "SHOULD NOT",

45 "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC

46 2119. The words 'must', 'can', and 'will' are forbidden.

47 2 Objects

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

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

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

63 2.1 Base Objects

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

66 2.1.1 Attribute

67 An Attribute object is a structure (see [Table 1](#)) used for sending and receiving Managed Object attributes.
68 The *Attribute Name* is a text-string that is used to identify the attribute. The *Attribute Index* is an index
69 number assigned by the key management server when a specified named attribute is allowed to have
70 multiple instances. The Attribute Index is used to identify the particular instance. Attribute Indices SHALL
71 start with 0. The Attribute Index of an attribute SHALL NOT change when other instances are added or
72 deleted. For example, if a particular attribute has 4 instances with Attribute Indices 0, 1, 2 and 3, and the
73 instance with Attribute Index 2 is deleted, then the Attribute Index of instance 3 is not changed. Attributes
74 that have a single instance have an Attribute Index of 0, which is assumed if the Attribute Index is not
75 specified. The *Attribute Value* is either a primitive data type or structured object, depending on the
76 attribute.

Deleted: Table 1

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

Deleted: Required

77 **Table 1: Attribute Object Structure**

78 2.1.2 Credential

79 A credential is a structure (see [Table 2](#)) used for client identification purposes and is not managed by the
80 key management system (e.g., user id/password pairs, Kerberos tokens, etc). See Section 8 .

Deleted: Table 2

Object	Encoding	<u>REQUIRED</u>
Credential	Structure	Yes
Credential Type	Enumeration	Yes
Credential Value	Octet String	Yes

Deleted: Required

81

Table 2: Credential Object Structure

82 2.1.3 Key Block

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

Deleted: Table 3

- 85 • *Raw* – This is a key that contains only cryptographic key material, encoded as a string of bytes.
- 86 • *Opaque* – This is an encoded key for which the encoding is unknown to the key management
 87 system. It is encoded as a string of bytes.
- 88 • *PKCS1* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#1 object.
- 89 • *PKCS8* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#8 object,
 90 supporting both RSAPrivateKey syntax and EncryptedPrivateKey.
- 91 • Several *Transparent Key* types – These are algorithm-specific structures containing defined
 92 values for the various key types, as defined in Section 2.1.7 .
- 93 • *Extensions* – These are vendor-specific extensions to allow for proprietary or legacy key formats.

94 The Key Block also contains the Cryptographic Algorithm and the Cryptographic Length of the key
 95 contained in the Key Value field. Some example values are:

- 96 • RSA keys are typically 1024, 2048 or 3072 bits in length
- 97 • 3DES keys are typically 168 bits in length
- 98 • AES keys are typically 128 or 256 bits in length

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

Object	Encoding	<u>REQUIRED</u>
Key Block	Structure	Yes
Key Value Type	Enumeration	Yes
Key Value	Octet String: for wrapped Key Value; Structure: for plaintext Key Value	Yes
Cryptographic Algorithm	Enumeration	Yes, MAY be omitted only if this information is available from the Key Value. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Length SHALL also be present.
Cryptographic Length	Integer	Yes, MAY be omitted only if this information is available from the Key Value. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Algorithm SHALL also be present.
Key Wrapping Data	Structure	No, SHALL only be present if the key is wrapped.

Deleted: Required

Table 3: Key Block Object Structure

2.1.4 Key Value

The *Key Value* is used only inside a Key Block and is either an Octet String or a structure (see [Table 4](#)):

Deleted: Table 4

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

Deleted: optional

Object	Encoding	<u>REQUIRED</u>
Key Value	Structure	Yes
Key Material	Octet String: for Raw, Opaque, PKCS1, PKCS8, or Vendor Extension Key Value types; Structure: for Transparent, or Vendor Extension Key Value Types	Yes
Attribute	Attribute Object, see Section 2.1.1	No. MAY be repeated

Deleted: Required

Table 4: Key Value Object Structure

111 **2.1.5 Key Wrapping Data**

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

Deleted: optional

Deleted: Table 5

115 This structure contains fields for:

- 116 • A *Wrapping Method*, which indicates the method used to wrap the Key Value.
- 117 • *Encryption Key Information*, which contains the Unique Identifier value of the encryption key and
 118 associated cryptographic parameters.
- 119 • *MAC/Signature Key Information*, which contains the Unique Identifier value of the MAC/signature
 120 key and associated cryptographic parameters.
- 121 • A *MAC/Signature*, which contains the MAC or signature of the Key Value.
- 122 • An *IV/Counter/Nonce*, if REQUIRED by the wrapping method.

Deleted: required

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

128 The following wrapping methods are currently defined:

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

Object	Encoding	<u>REQUIRED</u>
Key Wrapping Data	Structure	Yes
Wrapping Method	Enumeration	Yes
Encryption Key Information	Structure	No. Corresponds to the key that was used to encrypt the Key Value.
MAC/Signature Key Information	Structure	No. Corresponds to the symmetric key used to MAC the Key Value or the private key used to sign the Key Value
MAC/Signature	Octet String	No
IV/Counter/Nonce	Octet String	No

Deleted: Required

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

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

Deleted: Table 6

Deleted: Table 7

Object	Encoding	<u>REQUIRED</u>
Encryption Key Information	Structure	Yes
Unique Identifier	Text string	Yes
Cryptographic Parameters	Structure	No

Deleted: Required

Table 6: Encryption Key Information Object Structure

Object	Encoding	<u>REQUIRED</u>
MAC/Signature Key Information	Structure	Yes
Unique Identifier	Text string	Yes. It MAY be 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	No

Deleted: Required

Table 7: MAC/Signature Key Information Object Structure

2.1.6 Key Wrapping Specification

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

Deleted: Table 8

This structure contains:

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

Object	Encoding	<u>REQUIRED</u>
Key Wrapping Specification	Structure	Yes
Wrapping Method	Enumeration	Yes
Encryption Key Information	Structure	No
MAC/Signature Key Information	Structure	No
Attribute Name	Text String	No, MAY be repeated

Deleted: Required

Table 8: Key Wrapping Specification Object Structure

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

161 2.1.7 Transparent Key Structures

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

164 2.1.7.1 Transparent Symmetric Key

165 If the Key Value Type in the Key Block is *Transparent Symmetric Key*, then Key Material is a structure as
166 shown in [Table 9](#).

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
Key	Octet String	Yes

Deleted: Table 9

Deleted: Required

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

168 2.1.7.2 Transparent DSA Private Key

169 If the Key Value Type in the Key Block is *Transparent DSA Private Key*, then Key Material is a structure
170 as shown in [Table 10](#).

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
P	Big Integer	Yes
Q	Big Integer	Yes
G	Big Integer	Yes
X	Big Integer	Yes

Deleted: Table 10

Deleted: Required

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

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

174 2.1.7.3 Transparent DSA Public Key

175 If the Key Value Type in the Key Block is *Transparent DSA Public Key*, then Key Material is a structure as
176 shown in [Table 11](#).

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
P	Big Integer	Yes
Q	Big Integer	Yes
G	Big Integer	Yes
Y	Big Integer	Yes

Deleted: Table 11

Deleted: Required

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

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

180 **2.1.7.4 Transparent RSA Private Key**

181 If the Key Value Type in the Key Block is *Transparent RSA Private Key*, then Key Material is a structure
 182 as shown in [Table 12](#).

Deleted: Table 12

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

Deleted: Required

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

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

- 185 • Private Exponent
- 186 • P and Q (the first two prime factors of Modulus)
- 187 • Prime Exponent P and Prime Exponent Q.

188 **2.1.7.5 Transparent RSA Public Key**

189 If the Key Value Type in the Key Block is *Transparent RSA Public Key*, then Key Material is a structure as
 190 shown in [Table 13](#).

Deleted: Table 13

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
Modulus	Big Integer	Yes
Public Exponent	Big Integer	Yes

Deleted: Required

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

192 **2.1.7.6 Transparent DH Private Key**

193 If the Key Value Type in the Key Block is *Transparent DH Private Key*, then Key Material is a structure as
 194 shown in [Table 14](#).

Deleted: Table 14

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
P	Big Integer	Yes
G	Big Integer	Yes
Q	Big Integer	No
J	Big Integer	No
X	Big Integer	Yes

Deleted: Required

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

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

198 **2.1.7.7 Transparent DH Public Key**

199 If the Key Value Type in the Key Block is *Transparent DH Public Key*, then Key Material is a structure as
 200 shown in [Table 15](#).

Deleted: Table 15

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
P	Big Integer	Yes
G	Big Integer	Yes
Q	Big Integer	No
J	Big Integer	No
Y	Big Integer	Yes

Deleted: Required

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

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

204 **2.1.7.8 Transparent ECDSA Private Key**

205 If the Key Value Type in the Key Block is *Transparent ECDSA Private Key*, then Key Material is a
 206 structure as shown in [Table 16](#).

Deleted: Table 16

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
D	Big Integer	Yes

Deleted: Required

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

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

209 **2.1.7.9 Transparent ECDSA Public Key**

210 If the Key Value Type in the Key Block is *Transparent ECDSA Public Key*, then Key Material is a structure
 211 as shown in [Table 17](#).

Deleted: Table 17

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
Q String	Octet String	Yes

Deleted: Required

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

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

214 **2.1.7.10 Transparent ECDH Private Key**

215 If the Key Value Type in the Key Block is *Transparent ECDH Private Key*, then Key Material is a structure
 216 as shown in [Table 18](#).

Deleted: Table 18

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
D	Big Integer	Yes

Deleted: Required

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

218 **2.1.7.11 Transparent ECDH Public Key**

219 If the Key Value Type in the Key Block is *Transparent ECDH Public Key*, then Key Material is a structure
 220 as shown in [Table 19](#).

Deleted: Table 19

Object	Encoding	<u>REQUIRED</u>
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
Q String	Octet String	Yes

Deleted: Required

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

222 **2.1.8 Template-Attribute Structures**

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

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

Object	Encoding	<u>REQUIRED</u>
Template-Attribute, Common Template-Attribute, Private Key Template- Attribute, Public Key Template-Attribute	Structure	Yes
Name	Structure, see Section 3.2	No, MAY be repeated.
Attribute	Attribute Object, see Section 2.1.1	No, MAY be repeated

Deleted: Required

227 **Table 20: Template-Attribute Object Structure**

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

229 **2.2 Managed Objects**

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

233 **2.2.1 Certificate**

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

Object	Encoding	<u>REQUIRED</u>
Certificate	Structure	Yes
Certificate Type	Enumeration	Yes
Certificate Value	Octet String	Yes

Deleted: Required

235

Table 21: Certificate Object Structure

236 2.2.2 Symmetric Key

237 A Managed Cryptographic Object that is a symmetric key.

Object	Encoding	<u>REQUIRED</u>
Symmetric Key	Structure	Yes
Key Block	Structure	Yes

Deleted: Required

238

Table 22: Symmetric Key Object Structure

239 2.2.3 Public Key

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

Object	Encoding	<u>REQUIRED</u>
Public Key	Structure	Yes
Key Block	Structure	Yes

Deleted: Required

242

Table 23: Public Key Object Structure

243 2.2.4 Private Key

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

Object	Encoding	<u>REQUIRED</u>
Private Key	Structure	Yes
Key Block	Structure	Yes

Deleted: Required

245

Table 24: Private Key Object Structure

246 2.2.5 Split Key

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

Object	Encoding	<u>REQUIRED</u>
Split Key	Structure	Yes
Split Key Parts	Integer	Yes
Key Part Identifier	Integer	Yes
Split Key Threshold	Integer	Yes
Split Key Method	Enumeration	Yes
Prime Field Size	Big Integer	No, <u>REQUIRED</u> only if Split Key Method is Polynomial Sharing Prime Field.
Key Block	Structure	Yes

Deleted: Required

Deleted: required

Table 25: Split Key Object Structure

253

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

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

- 258 • When the Split Key Method is XOR, then the Key Material in the Key Value of the Key Block is of
 259 length L bits. The number of split keys is Split Key Parts (identical to Split Key Threshold), and
 260 the secret is reconstructed by XORing all of the parts.
- 261 • When the Split Key Method is Polynomial Sharing Prime Field, then secret sharing is performed
 262 in the field $GF(\text{Prime Field Size})$, represented as integers, where Prime Field Size is a prime
 263 bigger than 2^L .
- 264 • When the Split Key Method is Polynomial Sharing $GF(2^{16})$, then secret sharing is performed in
 265 the field $GF(2^{16})$. The Key Material in the Key Value of the Key Block is a bit string of length L ,
 266 and when L is bigger than 2^{16} , then secret sharing is applied piecewise in pieces of 16 bits each.
 267 The Key Material in the Key Value of the Key Block is the concatenation of the corresponding
 268 shares of all pieces of the secret.

269 Secret sharing is performed in the field $GF(2^{16})$, which is represented as an algebraic extension of
 270 $GF(2^8)$:

271 $GF(2^{16}) \approx GF(2^8)[y]/(y^2+y+m)$, where m is defined later.

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

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

276 $GF(2^8) \approx GF(2)[x]/(x^8+x^4+x^3+x+1)$.

277 An element of $GF(2^8)$ is represented as an octet. Addition and subtraction in $GF(2^8)$ is performed
 278 as a bit-wise XOR of the octets. Multiplication and inversion are more complex (see FIPS PUB
 279 197 Section 4.1 and 4.2 for details).

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

281 $m = x^5+x^4+x^3+x$,

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

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

285 $(ry + s)(uy + v) = ((r + s)(u + v) + sv)y + (ru + svm)$.

286 The inverse of an element $uy + v$ is given by
 287 $(uy + v)^{-1} = ud^{-1}y + (u + v)d^{-1}$, where $d = (u + v)v + mu^2$.

288 **2.2.6 Template**

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

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

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

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

301 Attributes applicable to objects created using the Template are:

- 302 • Cryptographic Algorithm
- 303 • Cryptographic Length
- 304 • Cryptographic Parameters
- 305 • Operation Policy Name
- 306 • Cryptographic Usage Mask
- 307 • Usage Limits
- 308 • Activation Date
- 309 • Process Start Date
- 310 • Protect Stop Date
- 311 • Deactivation Date
- 312 • Object Group
- 313 • Application Specific Information
- 314 • Contact Information
- 315 • Custom Attribute

Object	Encoding	<u>REQUIRED</u>
Template	Structure	Yes
Attribute	Attribute Object, see Section 2.1.1	Yes. MAY be repeated.

Deleted: Required

316 **Table 26: Template Object Structure**

317 **2.2.7 Secret Data**

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

Object	Encoding	<u>REQUIRED</u>
Secret Data	Structure	Yes
Secret Data Type	Enumeration	Yes
Key Block	Structure	Yes

Deleted: Required

321

Table 27: Secret Data Object Structure

322 2.2.8 Opaque Object

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

Object	Encoding	<u>REQUIRED</u>
Opaque Object	Structure	Yes
Opaque Data Type	Enumeration	Yes
Opaque Data Value	Octet String	Yes

Deleted: Required

325

Table 28: Opaque Object Structure

326 3 Attributes

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

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

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

338 The second table in each subsection lists certain attribute characteristics (e.g., "SHALL always have a
 339 value"). The "When implicitly set" characteristic indicates which operations (other than operations that
 340 manage attributes) are able to implicitly add to or modify the attribute of the object, which MAY be
 341 object(s) on which the operation is performed or object(s) created as a result of the operation. Implicit
 342 attribute changes MAY occur even if the attribute is not specified in the operation request itself.

343 3.1 Unique Identifier

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

Deleted: required

Deleted: recommended

Object	Encoding	REQUIRED
Unique Identifier	Text String	Yes

Deleted: Required

350 **Table 29: 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

351 **Table 30: Unique Identifier Attribute Rules**

352 3.2 Name

353 The *Name* attribute is a structure (see [Table 31](#)) used to identify and locate the object, assigned by the
 354 client, and that humans are able to interpret. The key management system MAY specify rules by which
 355 the client creates valid names. Clients are informed of such rules by a mechanism that is not specified by

Deleted: Table 31

356 | this standard. Names SHALL be unique within a given key management domain, but are not **REQUIRED**
 357 | to be globally unique.

Deleted: required

Object	Encoding	REQUIRED
Name	Structure	Yes
Name Value	Text String	Yes
Name Type	Enumeration	Yes

Deleted: Required

358 | **Table 31: 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

359 | **Table 32: Name Attribute Rules**

360 | 3.3 Object Type

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

Object	Encoding	REQUIRED
Object Type	Enumeration	Yes

Deleted: Required

363 | **Table 33: 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

364 | **Table 34: Object Type Attribute Rules**

365 | 3.4 Cryptographic Algorithm

366 | The cryptographic algorithm used by the object (e.g., RSA, DSA, DES, 3DES, AES, etc). This attribute
 367 | SHALL be set by the server when the object is created or registered and then SHALL NOT be changed.

Object	Encoding	<u>REQUIRED</u>
Cryptographic Algorithm	Enumeration	Yes

Deleted: Required

368

Table 35: 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

369

Table 36: Cryptographic Algorithm Attribute Rules

370 3.5 Cryptographic Length

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

Object	Encoding	<u>REQUIRED</u>
Cryptographic Length	Integer	Yes

Deleted: Required

374

Table 37: 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

375

Table 38: Cryptographic Length Attribute Rules

376 3.6 Cryptographic Parameters

377 The *Cryptographic Parameters* attribute is a structure (see [Table 39](#)) that contains a set of OPTIONAL
 378 fields that describe certain cryptographic parameters to be used when performing cryptographic
 379 operations using the object. It is possible that specific fields only pertain to certain types of Managed
 380 Cryptographic Objects.

Deleted: Table 39

Deleted: optional

Object	Encoding	<u>REQUIRED</u>
Cryptographic Parameters	Structure	Yes
Block Cipher Mode	Enumeration	No
Padding Method	Enumeration	No
Hashing Algorithm	Enumeration	No
Role Type	Enumeration	No

Deleted: Required

381

Table 39: 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

382

Table 40: Cryptographic Parameters Attribute Rules

383 Role Type definitions match those defined in ANSI X9 "TR-31 2005 Interoperable Secure Key Exchange
384 Key Block Specification for Symmetric Algorithms" and are defined in [Table 41](#);

Deleted: Table 41

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
KMOTH	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 41: Role Types

385
386 Accredited Standards Committee X9, Inc. - Financial Industry Standards (www.x9.org) contributed to
387 [Table 41](#). Key role names and descriptions are derived from material in the Accredited Standards
388 Committee X9, Inc's Technical Report "TR-31 2005 Interoperable Secure Key Exchange Key Block
389 Specification for Symmetric Algorithms" and used with the permission of Accredited Standards Committee
390 X9, Inc. in an effort to improve interoperability between X9 standards and OASIS KMIP. The complete
391 ANSI X9 TR-31 is available at www.x9.org.

Deleted: Table 41

392 3.7 Certificate Type

393 The type of a certificate (e.g., X.509, PGP, etc). This value SHALL be set by the server when the
394 certificate is created or registered and then SHALL NOT be changed.

Object	Encoding	<u>REQUIRED</u>
Certificate Type	Enumeration	Yes

Deleted: Required

Table 42: Certificate Type Attribute

395

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

Table 43: Certificate Type Attribute Rules

396

3.8 Certificate Issuer

397

398 | The Certificate Issuer attribute is a structure (see [Table 44](#)) used to provide identification of a certificate,
 399 containing the Issuer Distinguished Name (i.e., from the Issuer field of the certificate) and the Certificate
 400 Serial Number (i.e., from the Serial Number field of the certificate). This value SHALL be set by the server
 401 when the certificate is created or registered and then SHALL NOT be changed.

Deleted: Table 44

Object	Encoding	<u>REQUIRED</u>
Certificate Issuer	Structure	Yes
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)

Deleted: Required

Table 44: Certificate Issuer Attribute Structure

402

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

Table 45: Certificate Issuer Attribute Rules

403

3.9 Certificate Subject

404

405 | The Certificate Subject attribute is a structure (see [Table 46](#)) used to identify the subject of a certificate,
 406 containing the Subject Distinguished Name (i.e., from the Subject field of the certificate). It MAY include
 407 one or more alternative names (e.g., email address, IP address, DNS name) for the subject of the
 408 certificate (i.e., from the Subject Alternative Name extension within the certificate). These values SHALL
 409 be set by the server when the certificate is created or registered and SHALL NOT be changed until the
 410 certificate is renewed.

Deleted: Table 46

411 | If the Subject Alternative Name extension is included in the certificate and is marked *CRITICAL*, then it is
 412 possible to issue an X.509 certificate where the subject field is left blank. Therefore an empty string is an
 413 acceptable value for the Certificate Subject Distinguished Name.

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

Deleted: Required

414

Table 46: 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

415

Table 47: Certificate Subject Attribute Rules

416 3.10 Digest

417 The Digest attribute is a structure (see [Table 48](#)) that contains the digest value of the key or secret data
 418 (i.e., digest of the Key Material), certificate (i.e., digest of the Certificate Value), or opaque object (i.e.,
 419 digest of the Opaque Data Value). Multiple digests MAY be calculated using different algorithms. The
 420 mandatory digest SHALL be computed with the SHA-256 hashing algorithm; the server MAY store
 421 additional digests. The digest(s) are static and SHALL be generated by the server when the object is
 422 created or registered.

Deleted: Table 48

Object	Encoding	<u>REQUIRED</u>
Digest	Structure	Yes
Hashing Algorithm	Enumeration	Yes
Digest Value	Octet String	Yes

Deleted: Required

423

Table 48: Digest Attribute Structure

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

425 **3.11 Operation Policy Name**

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

Object	Encoding	<u>REQUIRED</u>
Operation Policy Name	Text String	Yes

Deleted: Required

434

Table 50: 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

435

Table 51: Operation Policy Name Attribute Rules

436 **3.11.1 Operations outside of operation policy control**

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

- 439 • Create
- 440 • Create Key Pair
- 441 • Register
- 442 • Certify
- 443 • Validate
- 444 • Query
- 445 • Cancel
- 446 • Poll

447 **3.11.2 Default Operation Policy**

448 A key management system implementation SHALL implement at least one named operation policy, which
 449 is used for objects when the *Operation Policy* attribute is not specified by the Client in a *Create* or
 450 *Register* operation, or in a template specified in these operations. This policy is named *default*. It specifies

451 the following rules for operations on objects created or registered with this policy, depending on the object
452 type.

453 **3.11.2.1 Default Operation Policy for Secret Objects**

454 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

455 **Table 52: Default Operation Policy for Secret Objects**

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

458 **3.11.2.2 Default Operation Policy for Certificates and Public Key Objects**

459 This policy applies to Certificates and Public Keys.

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

460

Table 53: Default Operation Policy for Certificates and Public Key Objects

461

3.11.2.3 Default Operation Policy for Template Objects

462
463
464
465
466

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

Default Operation Policy for Private Template Objects	
Operation	Policy
Locate	Allowed to 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

467

Table 54: Default Operation Policy for Private Template Objects

468 In addition to private template objects (which are controlled by the above policy, and which MAY be
469 created by clients or the server), publicly known and usable templates MAY be created and managed by
470 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

471 **Table 55: Default Operation Policy for Public Template Objects**

472 3.12 Cryptographic Usage Mask

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

- 475 • Sign
- 476 • Verify
- 477 • Encrypt
- 478 • Decrypt
- 479 • Wrap Key
- 480 • Unwrap Key
- 481 • Export
- 482 • MAC Generate
- 483 • MAC Verify
- 484 • Derive Key
- 485 • Content Commitment
- 486 • Key Agreement
- 487 • Certificate Sign
- 488 • CRL Sign
- 489 • Generate Cryptogram
- 490 • Validate Cryptogram
- 491 • Translate Encrypt
- 492 • Translate Decrypt
- 493 • Translate Wrap
- 494 • Translate Unwrap

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

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

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

500 **Table 56: X.509 Key Usage to Cryptographic Usage Mask Mapping**

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

Object	Encoding	<u>REQUIRED</u>
Cryptographic Usage Mask	Integer	Yes

Deleted: Required

505 **Table 57: 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

506 **Table 58: Cryptographic Usage Mask Attribute Rules**

507 3.13 Lease Time

508 The *Lease Time* attribute defines a time interval for a Managed Cryptographic Object that indicates how
 509 long a client SHOULD use the object. This attribute always holds the initial value of a lease, and not the

510 actual remaining time. Once the lease expires, then the client is only able to renew the lease by calling
 511 Obtain Lease. A server SHOULD store in this attribute the maximum Lease Time it is able to serve and a
 512 client obtains lease time (with Obtain Lease) that is less than or equal to the maximum Lease Time. This
 513 attribute is read-only for clients. It SHALL be modified by the server only.

Object	Encoding	<u>REQUIRED</u>
Lease Time	Interval	Yes

Deleted: Required

514 **Table 59: 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

515 **Table 60: Lease Time Attribute Rules**

516 **3.14 Usage Limits**

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

- 524 • *Usage Limits Total Bytes* – the total number of bytes allowed to be protected. This is the total
 525 value for the entire life of the object, and SHALL NOT be changed once the object begins to be
 526 used for protection purposes.
- 527 • *Usage Limits Byte Count* – the currently remaining number of bytes allowed to be protected.
- 528 • *Usage Limits Total Objects* – the total number of objects allowed to be protected. This is the total
 529 value for the entire life of the object, and SHALL NOT be changed once the object begins to be
 530 used for protection purposes.
- 531 • *Usage Limits Object Count* – the currently remaining number of objects allowed to be protected.

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

Object	Encoding	<u>REQUIRED</u>
Usage Limits	Structure	Yes
Usage Limits Total	Big Integer	No. SHALL be present if

Deleted: Required

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

538

Table 61: Usage Limits Attribute Structure

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

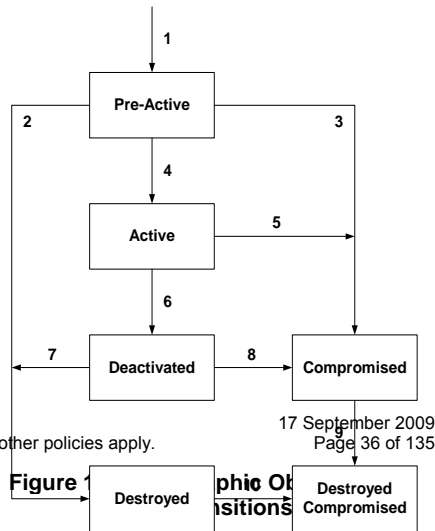
539

Table 62: Usage Limits Attribute Rules

540 **3.15 State**

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

- 546 • *Pre-Active*: The object exists but is not yet usable for
 547 any cryptographic purpose.
- 548 • *Active*: The object MAY be used for all cryptographic
 549 purposes that are allowed by its Cryptographic Usage
 550 Mask attribute.
- 551 • *Deactivated*: The object SHALL NOT be used for
 552 protection purpose (e.g., encryption or signing), but, if
 553 permitted by the Cryptographic Usage Mask attribute,
 554 then MAY be used for process purposes (e.g.,
 555 decryption or verification), but only under
 556 extraordinary circumstances and when special
 557 permission is granted.
- 558 • *Compromised*: It is possible that the object has been
 559 compromised, and SHOULD only be used for



560 process purposes in a client that is trusted to handle compromised cryptographic objects.

561 • *Destroyed*: The object is no longer usable for any purpose.

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

564 State transitions occur as follows:

565 1. The transition from a non-existent key to the Pre-Active state is caused by the creation of the
566 object. When an object is created or registered, it automatically goes from non-existent to Pre-
567 Active. If, however, the operation that creates or registers the object contains an Activation Date
568 that has already occurred, then the state immediately transitions to Active. In this case, the server
569 SHALL set the Activation Date attribute to the time when the operation is received, or fail the
570 request attempting to create or register the object, depending on server policy. If the operation
571 contains an Activation Date attribute in the future, or contains no Activation Date, then the
572 Cryptographic Object is initialized in the key management system in the Pre-Active state.

573 2. The transition from Pre-Active to Compromised is caused by a client issuing a Revoke operation
574 with a Revocation Reason of Compromised.

575 3. The transition from Pre-Active to Active SHALL occur in one of three ways:

576 • The object has an Activation Date in the future. At the time that the Activation Date is
577 reached, the server changes the state to Active.

578 • A client issues a Modify Attribute operation, modifying the Activation Date to a date in the
579 past, or the current date. In this case, the server SHALL either set the Activation Date
580 attribute to the date in the past or fail the operation, depending on server policy.

581 • A client issues an Activate operation on the object. The server SHALL set the Activation
582 Date to the time the Activate operation is received.

583 4. The transition from Active to Compromised is caused by a client issuing a Revoke operation with
584 a Revocation Reason of Compromised.

585 5. The transition from Active to Deactivated SHALL occur in one of three ways:

586 • The object's Deactivation Date is reached.

587 • A client issues a Revoke operation, with a Revocation Reason other than Compromised.

588 • The client issues a Modify Attribute operation, modifying the Deactivation Date to a date in
589 the past, or the current date. In this case, the server SHALL either set the Deactivation
590 Date attribute to the date in the past or fail the operation, depending on server policy.

591 6. The transition from Deactivated to Destroyed is caused by a client issuing a Destroy operation.
592 The server destroys the object when (and if) server policy dictates.

593 7. The transition from Deactivated to Compromised is caused by a client issuing a Revoke operation
594 with a Revocation Reason of Compromised.

595 8. The transition from Compromised to Destroyed Compromised is caused by a client issuing a
596 Destroy operation. The server destroys the object when (and if) server policy dictates.

597 9. The transition from Destroyed to Destroyed Compromised is caused by a client issuing a Revoke
598 operation with a Revocation Reason of Compromised.

599 Only the transitions described above are permitted.

Object	Encoding	<u>REQUIRED</u>
State	Enumeration	Yes

Deleted: Required

600 **Table 63: State Attribute**

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

Table 64: State Attribute Rules

601

602 3.16 Initial Date

603 This is the date and time when the Managed Object was first created or registered at the server. This time
604 corresponds to state transition 1 (see Section 3.15). This attribute SHALL be set by the server when the
605 object is created or registered, and then SHALL NOT be changed. This attribute is also set for non-
606 cryptographic objects (e.g., templates) when they are first registered with the server.

Object	Encoding	REQUIRED
Initial Date	Date-Time	Yes

Deleted: Required

Table 65: Initial Date Attribute

607

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

Table 66: Initial Date Attribute Rules

608

609 3.17 Activation Date

610 This is the date and time when the Managed Cryptographic Object MAY begin to be used. This time
611 corresponds to state transition 4 (see Section 3.15). The object SHALL NOT be used for any
612 cryptographic purpose before the *Activation Date* has been reached. Once the state transition has
613 occurred, then this attribute SHALL NOT be modified by the server or client.

Object	Encoding	REQUIRED
Activation Date	Date-Time	Yes

Deleted: Required

Table 67: Activation Date Attribute

614

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

Table 68: Activation Date Attribute Rules

615

3.18 Process Start Date

616

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

Object	Encoding	<u>REQUIRED</u>
Process Start Date	Date-Time	Yes

Deleted: Required

623

Table 69: Process Start Date Attribute

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

Table 70: Process Start Date Attribute Rules

624

3.19 Protect Stop Date

625

626 This is the date and time when a Managed Symmetric Key Object SHALL NOT be used for protect
627 purposes (e.g., encryption or wrapping), depending on the value of its Cryptographic Usage Mask
628 attribute. This value MAY be equal to, but SHALL NOT be later than the Deactivation Date. Once the
629 *Protect Stop Date* has occurred, then this attribute SHALL NOT be modified by the server or the client.

Object	Encoding	<u>REQUIRED</u>
Protect Stop Date	Date-Time	Yes

Deleted: Required

630

Table 71: Protect Stop Date Attribute

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

631

Table 72: Protect Stop Date Attribute Rules

632 3.20 Deactivation Date

633 This is the date and time when the Managed Cryptographic Object SHALL NOT be used for any purpose,
 634 except for decryption, signature verification, or unwrapping, but only under extraordinary circumstances
 635 and only when special permission is granted. This time corresponds to state transition 6 (see Section
 636 3.15). Once this transition has occurred, then this attribute SHALL NOT be modified by the server or
 637 client.

Object	Encoding	<u>REQUIRED</u>
Deactivation Date	Date-Time	Yes

Deleted: Required

638

Table 73: Deactivation Date Attribute

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

639

Table 74: Deactivation Date Attribute Rules

640 **3.21 Destroy Date**

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

Object	Encoding	<u>REQUIRED</u>
Destroy Date	Date-Time	Yes

Deleted: Required

644 **Table 75: 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

645 **Table 76: Destroy Date Attribute Rules**

646 **3.22 Compromise Occurrence Date**

647 This is the date and time when the Managed Cryptographic Object was first believed to be compromised.
 648 If it is not possible to estimate when the compromise occurred, then this value SHOULD be set to the
 649 Initial Date for the object.

Object	Encoding	<u>REQUIRED</u>
Compromise Occurrence Date	Date-Time	Yes

Deleted: Required

650 **Table 77: 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

651 **Table 78: Compromise Occurrence Date Attribute Rules**

652 **3.23 Compromise Date**

653 This is the date and time when the Managed Cryptographic Object entered into the compromised state.
 654 This time corresponds to state transitions 3, 5, 8, or 10 (see Section 3.15). This time indicates when the
 655 key management system was made aware of the compromise, not necessarily when the compromise

656 occurred. This attribute is set by the server when it receives a Revoke operation with a Revocation
 657 Reason of Compromised, or due to server policy or out-of-band administrative action.

Object	Encoding	<u>REQUIRED</u>
Compromise Date	Date-Time	Yes

Deleted: Required

658 **Table 79: 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

659 **Table 80: Compromise Date Attribute Rules**

660 3.24 Revocation Reason

661 The Revocation Reason attribute is a structure (see [Table 81](#)) used to indicate why the Managed
 662 Cryptographic Object was revoked (e.g., “compromised”, “expired”, “no longer used”, etc). This attribute is
 663 only changed by the server as a part of the Revoke Operation.

Deleted: Table 81

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

Deleted: optional

Object	Encoding	<u>REQUIRED</u>
Revocation Reason	Structure	Yes
Revocation Reason Code	Enumeration	Yes
Revocation Message	Text String	No

Deleted: Required

667 **Table 81: 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

668 **Table 82: Revocation Reason Attribute Rules**

669 **3.25 Archive Date**

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

Object	Encoding	<u>REQUIRED</u>
Archive Date	Date-Time	Yes

Deleted: Required

673 **Table 83: Archive Date Attribute**

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

674 **Table 84: Archive Date Attribute Rules**

675 **3.26 Object Group**

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

Object	Encoding	<u>REQUIRED</u>
Object Group	Text String	Yes

Deleted: Required

678 **Table 85: 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

679 **Table 86: Object Group Attribute Rules**

680 **3.27 Link**

681 The Link attribute is a structure (see [Table 87](#)) used to create a link from one Managed Cryptographic
 682 Object to another, closely related target Managed Cryptographic Object. The link has a type, and the
 683 allowed types differ, depending on the Object Type of the Managed Cryptographic Object. The *Linked*
 684 *Object Identifier* identifies the target Managed Cryptographic Object by its Unique Identifier. The link

Deleted: Table 87

685 contains information associated between the Managed Cryptographic Objects (e.g., the private key
 686 corresponding to a public key; the parent certificate for a certificate in a chain; or for a derived symmetric
 687 key, the base key from which it was derived).

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

- 690 • *Private Key Link*. For a Public Key object: the private key corresponding to the public key
- 691 • *Public Key Link*. For a Private Key object: the public key corresponding to the private key. For a
 692 Certificate object: the public key certified by the certificate
- 693 • *Certificate Link*. For Certificate objects: the parent certificate for a certificate in a certificate chain.
 694 For Public Key objects: the corresponding certificate(s), containing the same public key
- 695 • *Derivation Base Object Link* for a derived Symmetric Key object: the object(s) from which the
 696 current symmetric key was derived
- 697 • *Derived Key Link*: the symmetric key(s) that were derived from the current object.
- 698 • *Replacement Object Link*. For a Symmetric Key, Private Key, or Public Key object: the key that
 699 resulted from the re-key of the current key. For a Certificate object: the certificate that resulted
 700 from the re-certify. Note that there SHALL be only one such replacement object.
- 701 • *Replaced Object Link*. For a Symmetric Key, Private Key, or Public Key object: the key that was
 702 re-keyed to obtain the current key. For a Certificate object: the certificate that was re-certified to
 703 obtain the current certificate

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

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

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

Object	Encoding	<u>REQUIRED</u>
Link	Structure	Yes
Link Type	Enumeration	Yes
Linked Object Identifier	Text String	Yes

Deleted: Required

714 **Table 87: 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

715

Table 88: Link Attribute Structure Rules

716 3.28 Application Specific Information

717 | The Application Specific Information attribute is a structure (see [Table 89](#)) used to store data specific to
718 the application(s) using the Managed Object. It consists of the following fields: an *Application Namespace*
719 and *Application Data* specific to that application namespace. A list of standard application namespaces is
720 provided in [TBD].

Deleted: Table 89

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

727

Object	Encoding	<u>REQUIRED</u>
Application Specific Information	Structure	Yes
Application Namespace	Text String	Yes
Application Data	Text String	Yes

Deleted: Required

728

Table 89: Application Specific Information Attribute

729

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

730

Table 90: Application Specific Information Attribute Rules

731 3.29 Contact Information

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

Deleted: optional

Object	Encoding	<u>REQUIRED</u>
Contact Information	Text String	Yes

Deleted: Required

734

Table 91: 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

735

Table 92: Contact Information Attribute Rules

736 **3.30 Last Changed Date**

737 This is a meta attribute that contains the date and time of the last change to the contents or attributes of
738 the specified object.

Object	Encoding	REQUIRED
Last Changed Date	Date-Time	Yes

Deleted: Required

739

Table 93: Last Changed 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

740

Table 94: Last Changed Date Attribute Rules

741 **3.31 Custom Attribute**

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

Object	Encoding	<u>REQUIRED</u>
Custom Attribute	Any data type or structure	Yes. The name of the attribute SHALL start with 'x-' or 'y-'.

Deleted: Required

749

Table 95 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

750

Table 96: Custom Attribute Rules

751

4 Client-to-Server Operations

752
753
754
755
756
757

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

758
759

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

760
761
762
763
764

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

765
766
767
768
769
770
771

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

772
773
774
775
776

Requests MAY contain attribute values to be assigned to the object. This information is specified with a Template-Attribute (see Section 2.1.8) that contains zero or more template names and zero or more individual attributes. If more than one template name is specified, and there is a conflict between the single-instance attributes in the templates, then the value in the subsequent template takes precedence. If there is a conflict between the single-instance attributes in the request and the single-instance attributes

777 in a specified template, then the attribute values in the request take precedence. For multi-value
 778 attributes, the union of attribute values is used when the attributes are specified more than once.

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

Deleted: optionally

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

785 **4.1 Create**

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

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

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

Request Payload		
Object	<u>REQUIRED</u>	Description
Object Type	Yes	Determines the type of object to be created.
Template-Attribute	Yes	Specifies desired object attributes using templates and/or as individual attributes.

Formatted Table

Deleted: Required

793 **Table 97: Create Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Object Type	Yes	Type of object created.
Unique Identifier	Yes	The Unique Identifier of the newly created object.
Template-Attribute	No	An <u>OPTIONAL</u> list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Formatted Table

Deleted: Required

Deleted: optional

794 **Table 98: Create Response Payload**

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

Attribute	<u>REQUIRED</u>
Cryptographic Algorithm	Yes
Cryptographic Usage Mask	Yes

Deleted: Required

797 **Table 99: Create Attribute Requirements**

798 **4.2 Create Key Pair**

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

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

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

Request Payload		
Object	<u>REQUIRED</u>	Description
Common Template-Attribute	No	Specifies desired attributes in templates and/or as individual attributes that apply to both the Private and Public Key Objects.
Private Key Template-Attribute	No	Specifies templates and/or attributes that apply to the Private Key Object. Order of precedence applies.
Public Key Template-Attribute	No	Specifies templates and/or attributes that apply to the Public Key Object. Order of precedence applies.

Deleted: Required
Formatted Table

810 **Table 100: Create Key Pair Request Payload**

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

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

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

Response Payload		
Object	<u>REQUIRED</u>	Description
Private Key Unique Identifier	Yes	The Unique Identifier of the newly created Private Key object.
Public Key Unique Identifier	Yes	The Unique Identifier of the newly created Public Key object.
Private Key Template-Attribute	No	An <u>OPTIONAL</u> 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.

Deleted: Required
Formatted Table

Deleted: optional

Public Key Template-Attribute	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.
-------------------------------	----	--

Deleted: optional

820 **Table 101: Create Key Pair Response Payload**

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

Attribute	REQUIRED	SHALL contain the same value for both Private and Public Key
Cryptographic Algorithm	Yes	Yes
Cryptographic Length	Yes	Yes
Cryptographic Usage Mask	Yes	No
Cryptographic Parameters	No	Yes

Deleted: Required

823 **Table 102: Create Key Pair Attribute Requirements**

824 **4.3 Register**

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

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

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

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

Deleted: Required

Formatted Table

836 **Table 103: Register Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the newly registered object.
Template-Attribute	No	An <u>OPTIONAL</u> list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Formatted Table
 Deleted: Required
 Deleted: optional

837 **Table 104: Register Response Payload**
 838 If a Managed Cryptographic Object is registered, then the following attributes SHALL be included in the
 839 Register request, either explicitly, or via specification of a template that contains the attribute.

Attribute	<u>REQUIRED</u>
Cryptographic Algorithm	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	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	Yes.

Deleted: Required

840 **Table 105: Register Attribute Requirements**

841 **4.4 Re-key**

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

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

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

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

851 If Offset is set and if such times exist, then the times of the new key SHALL be set based on the times of
 852 the existing key as follows:

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

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

853

Table 106: Computing New Dates from Offset during Re-key

854

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

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

855

Table 107: Re-key Attribute Requirements

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

Deleted: Required

Formatted Table

856

Table 108: Re-key Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the new Symmetric Key.
Template-Attribute	No	An <u>OPTIONAL</u> list of object attributes with values that were not specified in the request, but have been implicitly set by the key management server.

Deleted: Required

Formatted Table

Deleted: optional

857

Table 109: Re-key Response Payload

858 4.5 Derive Key

859 This request is used to derive a symmetric key using a key or secret data that is already known to the key
 860 management system. It SHALL only apply to Managed Cryptographic Objects that have the Derive Key
 861 bit set in the Cryptographic Usage Mask attribute of the specified Managed Object (i.e., are able to be
 862 used for key derivation). If the operation is issued for an object that does not have this bit set, then the
 863 server SHALL return a response with a Result Reason of Operation Not Supported. For all derivation
 864 methods, the client SHALL specify the desired length of the derived key or secret using the Cryptographic
 865 Length attribute. If a key is created, then the client SHALL specify both its Cryptographic Length and
 866 Cryptographic Algorithm. If the specified length exceeds the output of the derivation method, then the
 867 server SHALL return an error. Clients have the option to derive multiple keys and IVs by creating a Secret
 868 Data object and specifying a Cryptographic Length that is the total length of the derived object. The length
 869 SHALL NOT exceed the length of the output returned by the chosen derivation method.

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

- 874 • *PBKDF2* – This method is used to derive a symmetric key from a password or pass phrase. The
 875 PBKDF2 method is published in RSA Laboratories' Public-Key Cryptography Standards (PKCS)
 876 series, specifically PKCS #5 v2.0, and also published as Internet Engineering Task Force's RFC
 877 2898.
- 878 • *HASH* – This method derives a key by computing a hash over the derivation key or the derivation
 879 data.
- 880 • *HMAC* – This method derives a key by computing an HMAC over the derivation data.

- 881 • *ENCRYPT* – This method derives a key by encrypting the derivation data.
- 882 • *NIST800-108-C* – This method derives a key by computing the KDF in Counter Mode as specified
- 883 in NIST SP 800-108.
- 884 • *NIST800-108-F* – This method derives a key by computing the KDF in Feedback Mode as
- 885 specified in NIST SP 800-108.
- 886 • *NIST800-108-DPI* – This method derives a key by computing the KDF in Double-Pipeline Iteration
- 887 Mode as specified in NIST SP 800-108.
- 888 • *Extensions*

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

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

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

Formatted Table
 Deleted: Required

Deleted: required

895 **Table 110: Derive Key Request Payload**

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

Formatted Table

Deleted: Required

Deleted: optional

896

Table 111: Derive Key Response Payload

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

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

Deleted: Required

899

Table 112: Derivation Parameters Structure (Except PBKDF2)

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

Deleted: required

Deleted: required

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

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

Deleted: optional

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

917 The PBKDF2 derivation method requires two additional parameters:

Object	Encoding	<u>REQUIRED</u>
Derivation Parameters	Structure	Yes
Cryptographic Parameters	Structure	No, depends on the PRF.

Deleted: Required

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

918 **Table 113: PBKDF2 Derivation Parameters Structure**

919 **4.6 Certify**

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

Deleted: optional

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

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

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

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

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

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

Formatted Table

Deleted: Required

933 **Table 114: Certify Request Payload**

Response Payload		
Object	REQUIRED	Description
Unique Identifier	Yes	The Unique Identifier of the new certificate.
Template-Attribute	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.

Formatted Table

Deleted: Required

Deleted: optional

934

Table 115: Certify Response Payload

935 **4.7 Re-certify**

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

Deleted: optional

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

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

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

945 Since the new certificate assumes the name attribute of the existing certificate, Re-certify SHOULD only
 946 be performed once on a given certificate.

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

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

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

953

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

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

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

955

Table 117: Re-certify Attribute Requirements

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

Formatted Table

Deleted: Required

956

Table 118: Re-certify Request Payload

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

Deleted: Required
Formatted Table
Deleted: optional

957 **Table 119: Re-certify Response Payload**

958 **4.8 Locate**

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

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

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

Deleted: required

976 When using the Name or Object Group attributes for identification, wild-cards or regular expressions MAY
 977 be supported by specific key management system implementations.

978 The Date attributes (e.g., Initial Date, Activation Date, etc) are used to specify a time or a time range. If a
 979 single instance of a given Date attribute is used (e.g., the Activation Date), then objects with the same
 980 Date attribute are matching candidate objects. If two instances of the same Date attribute are used (i.e.,
 981 with two different values specifying a range), then objects for which the Date attribute is inside or at a limit
 982 of the range are matching candidate objects. If a Date attribute is set to its largest possible value, then it
 983 is equivalent to an undefined attribute.

984 When the Cryptographic Usage Mask attribute is specified in the request, candidate objects are
 985 compared against this field via an operation that consists of a logical AND of the requested mask with the
 986 mask in the candidate object, and then a comparison of the resulting value with the requested mask. For
 987 example, if the request contains a mask value of 10001100010000, and a candidate object mask contains
 988 10000100010000, then the logical AND of the two masks is 10000100010000, which is compared against
 989 10001100010000 and fails the match. This means that a matching candidate object at least has all of the
 990 bits set in its mask that are set in the requested mask, but MAY have additional bits set.

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

994 When an attribute defined as a structure is specified, all of the structure fields are not **REQUIRED** to be
 995 specified. For instance, for the Link attribute, if the Linked Object Identifier value is specified without the
 996 Link Type value, then matching candidate objects have the Linked Object Identifier as specified,
 997 irrespective of their Link Type.

Deleted: required

998 The Storage Status Mask field (see Section 9.1.3.3.2) is used to indicate whether only on-line objects,
 999 only archived objects, or both on-line and archived objects are to be searched. Note that the server MAY
 1000 store attributes of archived objects in order to expedite Locate operations that search through archived
 1001 objects.

Request Payload		
Object	<u>REQUIRED</u>	Description
Maximum Items	No	An Integer object that indicates the maximum number of object identifiers the server SHALL return.
Storage Status Mask	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	Yes, MAY be repeated	Specifies an attribute and its value that are <u>REQUIRED</u> to match the desired object.

Deleted: Required
 Formatted Table

Deleted: required

1002 **Table 120: Locate Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No, <u>MAY</u> be repeated	The Unique Identifier of the located objects.

Formatted Table
 Deleted: Required
 Deleted: May

1003 **Table 121: Locate Response Payload**

1004 4.9 Check

1005 This operation requests that the server checks for the use of a Managed Object according to values
 1006 specified in the request. This operation SHOULD only be used when placed in a batched set of
 1007 operations, usually following a Locate, Create, Create Pair, Derive Key, Certify, Re-Certify or Re-Key
 1008 operation, and followed by a Get operation. The Unique Identifier field in the request MAY be omitted if
 1009 the operation is in a batched set of operations and follows an operation that sets the ID Placeholder
 1010 variable.

1011 If the server determines that the client is allowed to use the object according to the specified attributes,
 1012 then the server returns the Unique Identifier of the object. If the server determines that the client is not
 1013 allowed to use the object according to the specified attributes, then the server invalidates the ID
 1014 Placeholder value and does not return the Unique Identifier, and the operation returns the set of attributes
 1015 specified in the request that caused the server policy denial. The only attributes returned are those
 1016 according to which the server determined that the client is not allowed to use the object, allowing the
 1017 client to determine how to proceed. The operation also returns a failure, and the server SHALL ignore any
 1018 subsequent operations in the batch.

1019 The additional objects that MAY be specified in the request are limited to:

- 1020 • Usage Limits Byte Count or Usage Limits Object Count (see Section 3.14)– The request MAY
 1021 contain the usage amount that the client deems necessary to complete its needed function. This
 1022 does not require that any subsequent Get Usage Allocation operations request this amount. It
 1023 only means that the client is ensuring that the amount specified is available.
- 1024 • Cryptographic Usage Mask – This is used to specify the cryptographic operations for which the
 1025 client intends to use the object (see Section 3.12). This allows the server to determine if the
 1026 policy allows this client to perform these operations with the object. Note that this MAY be a

1027 different value from the one specified in a *Locate* operation that precedes this operation. *Locate*,
 1028 for example, MAY specify a Cryptographic Usage Mask requesting a key that MAY be used for
 1029 both Encryption and Decryption, but the value in the Check operation MAY specify that the client
 1030 is only using the key for Encryption at this time.

- 1031 • Lease Time – This specifies a desired lease time (see Section 3.13). The client MAY use this to
 1032 determine if the server allows the client to use the object with the specified lease or longer.
 1033 Including this attribute in the Check operation does not actually cause the server to grant a lease,
 1034 but only indicates that the requested lease time value MAY be granted if requested by a
 1035 subsequent, batched, Obtain Lease operation.

1036 Note that these objects are not encoded in an Attribute structure as shown in Section 2.1.1

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object being checked. If omitted, then the ID Placeholder is substituted by the server.
Usage Limits Byte Count	No	Specifies the number of bytes to be protected to be checked against server policy. SHALL only be present if Usage Limits Object Count is not present.
Usage Limits Object Count	No	Specifies the number of objects to be protected to be checked against server policy. SHALL only be present if Usage Limits Byte Count is not present.
Cryptographic Usage Mask	No	Specifies the Cryptographic Usage for which the client uses the object.
Lease Time	No	Specifies a Lease Time value that the Client is asking the server to validate against server policy.

Formatted Table
 Deleted: Required

1037 **Table 122: Check Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object.
Usage Limits Byte Count	No	Returned by the Server if the Usage Limits value specified in the Request Payload is larger than the value that the server policy allows. SHALL only be present if Usage Limits Object Count is not present.
Usage Limits Object Count	No	Returned by the Server if the Usage Limits value specified in the Request Payload is larger than the value that the server policy allows. SHALL only be present if Usage Limits Byte Count is not present.
Cryptographic Usage Mask	No	Returned by the Server if the Cryptographic Usage Mask specified in the Request Payload is rejected by the server for policy violation.

Deleted: Required
 Formatted Table

Lease Time	No	Returned by the Server if the Lease Time value in the Request Payload is larger than a valid Lease Time the server MAY grant.
------------	----	---

Table 123: Check Response Payload

1038

1039 The encodings of the Usage limits Byte and Object Counts is as shown in Section 3.14 .

1040 **4.10 Get**

1041 This operation requests that the server returns the Managed Object specified in the request by its Unique
 1042 Identifier. The Unique Identifier field in the request MAY be omitted if the *Get* operation is in a batched set
 1043 of operations and follows an operation that sets the ID Placeholder variable.

1044 Only a single object is returned. The response contains the Unique Identifier of the object, along with the
 1045 object itself, which MAY be wrapped using a wrapping key specified in the request.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object being requested. If omitted, then the ID Placeholder is substituted by the server.
Key Wrapping Specification	No	Specifies keys and other information for wrapping the returned object. This field SHALL NOT be specified if the requested object is a Template.

Formatted Table

Deleted: Required

Table 124: Get Request Payload

1046

Response Payload		
Object	<u>REQUIRED</u>	Description
Object Type	Yes	Type of object
Unique Identifier	Yes	The Unique Identifier of the object
Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template, Secret Data, or Opaque Object	Yes	The cryptographic object being returned

Deleted: Required

Formatted Table

Table 125: Get Response Payload

1047

1048 **4.11 Get Attributes**

1049 This operation returns one or more attributes of a Managed Object. The object is specified by its Unique
 1050 Identifier and the attributes are specified by name in the request. If a specified attribute has multiple
 1051 instances, then all instances are returned. If a specified attribute does not exist (i.e., has no value), then it
 1052 SHALL NOT be present in the returned response. If no requested attributes exist, then the response
 1053 SHALL consist only of the Unique Identifier.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object whose attributes are being requested. If omitted, then the ID Placeholder is substituted by the server.
Attribute Name	Yes, <u>MAY</u> be repeated	Specifies a desired attribute of the object

Formatted Table
Deleted: Required

Deleted: May

1054 Table 126: Get Attributes Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	No, <u>MAY</u> be repeated	The requested attribute for the object

Formatted Table
Deleted: Required

Deleted: May

1055 Table 127: Get Attributes Response Payload

1056 **4.12 Get Attribute List**

1057 This operation returns a list of the attribute names associated with a Managed Object. The object is
1058 specified by its Unique Identifier.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object whose attribute names are being requested. If omitted, then the ID Placeholder is substituted by the server.

Deleted: Required
Formatted Table

1059 Table 128: Get Attribute List Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute Name	Yes, <u>MAY</u> be repeated	The requested attribute names for the object

Formatted Table
Deleted: Required

Deleted: May

1060 Table 129: Get Attribute List Response Payload

1061 **4.13 Add Attribute**

1062 This request adds a new attribute instance to a Managed Object and sets its value. The request contains
1063 the Unique Identifier of the Managed Object to which the attribute pertains, and the attribute name and
1064 value. For non multi-instance attributes, this is how they are created. For multi-instance attributes, this is
1065 how the first and subsequent values are created. Existing attribute values are only able to be changed by
1066 the Modify Attribute operation. Read-Only attributes are not able to be added using the Add Attribute
1067 operation. No Attribute Index SHALL be specified in the request. The response returns a new Attribute
1068 Index if the attribute being added is allowed to have multiple instances. Multiple Add Attribute requests
1069 MAY be included in a single batched request to add multiple attributes.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	The Unique Identifier of the object. If omitted, then the ID Placeholder is substituted by the server.
Attribute	Yes	Specifies the attribute of the object to be added.

Deleted: Required

Formatted Table

Table 130: Add Attribute Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	Yes	The added attribute

Deleted: Required

Formatted Table

Table 131: Add Attribute Response Payload

4.14 Modify Attribute

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

Deleted: optional

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

Formatted Table

Deleted: Required

Table 132: Modify Attribute Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	Yes	The modified attribute

Formatted Table

Deleted: Required

Table 133: Modify Attribute Response Payload

4.15 Delete Attribute

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

Deleted: Required

1089 instances, then the operation is rejected. Note that only a single attribute SHALL be deleted at a time.
 1090 Multiple delete operations (e.g., possibly batched) are necessary to delete several attributes. Attempting
 1091 to delete a non-existent attribute or using a non-existent Attribute Index in a delete operation SHALL
 1092 result in an error.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object whose attributes are being deleted. If omitted, then the ID Placeholder is substituted by the server.
Attribute Name	Yes	Specifies the name of the attribute to be deleted.
Attribute Index	No	Specifies the Index of the Attribute.

Deleted: Required
 Formatted Table

1093 **Table 134: Delete Attribute Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	Yes	The deleted attribute

Deleted: Required
 Formatted Table

1094 **Table 135: Delete Attribute Response Payload**

1095 **4.16 Obtain Lease**

1096 This request is used to obtain a new *Lease Time* for a specified Managed Object. The Lease Time is an
 1097 interval value that determines when the client's internal cache of information about the object expires and
 1098 needs to be renewed. If the returned value of the lease time is zero, then the server is indicating that no
 1099 lease interval is effective, and the client MAY use the object without any lease time limit. If a client's lease
 1100 expires, then the client SHALL NOT use the associated cryptographic object until a new lease is
 1101 obtained. If the server determines that a new lease SHALL NOT be issued for the specified cryptographic
 1102 object, then the server SHALL respond to the Obtain Lease request with a failure.

1103 The response payload for the operation also contains the current value of the Last Changed Date
 1104 attribute for the object. This MAY be used by the client to determine if any of the attributes cached by the
 1105 client need to be refreshed, by comparing this time to the time when the attributes were previously
 1106 obtained.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object for which the lease is being obtained. If omitted, then the <i>ID Placeholder</i> is substituted by the server.

Formatted Table
 Deleted: Required

1107 **Table 136: Obtain Lease Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object.
Lease Time	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 Changed Date	Yes	The date and time indicating when the latest change was made to the contents or any attribute of the specified object.

Deleted: Required
Formatted Table

1108 **Table 137: Obtain Lease Response Payload**

1109 **4.17 Get Usage Allocation**

1110 This request is used to obtain an allocation from the current Usage Limits values to allow the client to use
 1111 the Managed Cryptographic Object for protection purposes. It only applies to Managed Cryptographic
 1112 Objects that are able to be used for protection purposes (i.e., symmetric keys, private keys and public
 1113 keys) and is only valid if the Managed Cryptographic Object has a Usage Limits attribute. Usage for
 1114 process purposes (e.g., decryption, verification, etc.) is not limited and is not able to be allocated. A
 1115 Managed Cryptographic Object that has a Usage Limits attribute SHALL NOT be used by a client for
 1116 protection purposes unless an allocation has been obtained using this operation. The operation SHALL
 1117 only be requested during the time that protection is enabled for these objects (i.e., after the Activation
 1118 Date and before the Protect Stop Date). If the operation is requested for an object that has no Usage
 1119 Limits attribute, or is not an object that MAY be used for protection purposes, then the server SHALL
 1120 return a response with a Result Reason of Operation Not Supported.

1121 The fields in the request specify the number of bytes or number of objects that the client needs to protect.
 1122 Exactly one of the two count fields SHALL be specified in the request. If the requested amount is not
 1123 available or if the Managed Object is not able to be used for protection purposes at this time, then the
 1124 server SHALL return an error. The server SHALL assume that the entire allocated amount has been
 1125 consumed. Once the entire allocated amount has been consumed, the client SHALL NOT continue to use
 1126 the Managed Cryptographic Object for protection purposes until a new allocation is obtained.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object whose usage allocation is being requested. If omitted, then the ID Placeholder is substituted by the server.
Usage Limits Byte Count	No	The number of bytes to be protected. SHALL only be present if Usage Limits Object Count is not present.
Usage Limits Object Count	No	The number of objects to be protected. SHALL only be present if Usage Limits Byte Count is not present.

Deleted: Required
Formatted Table

1127 **Table 138: Get Usage Allocation Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object.

Formatted Table
Deleted: Required

1128

Table 139: Get Usage Allocation Response Payload

1129 The encodings of the Usage Limits Byte and Object Counts is as shown in Section 3.14 .

1130 4.18 Activate

1131 This request is used to activate a Managed Cryptographic Object. The request SHALL NOT specify a
1132 Template object. The request contains the Unique Identifier of the Managed Cryptographic Object. The
1133 operation is only able to be performed on an object in the Pre-Active state and has the effect of changing
1134 its state to Active, and setting its Activation Date to the current date and time.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object being activated. If omitted, then the ID Placeholder is substituted by the server.

Deleted: Required
Formatted Table

1135

Table 140: Activate Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object

Deleted: Required
Formatted Table

1136

Table 141: Activate Response Payload

1137 4.19 Revoke

1138 This request is used to revoke a Managed Cryptographic Object or an Opaque Object. The request
1139 SHALL NOT specify a Template object. The request contains the unique identifier of the Managed
1140 Cryptographic Object and a reason for the revocation (e.g., "compromised", "no longer used", etc).
1141 Special authentication and authorization SHOULD be enforced to perform this request (see Usage
1142 Guide). Only the object creator or an authorized security officer SHOULD be allowed to issue this
1143 request. The operation has one of two effects. If the revocation reason is "compromised", then the object
1144 is placed into the "compromised" state, and the Compromise Date attribute is set to the current date and
1145 time. Otherwise, the object is placed into the "deactivated" state, and the Deactivation Date attribute is set
1146 to the current date and time.

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object being revoked. If omitted, then the ID Placeholder is substituted by the server.
Revocation Reason	Yes	Specifies the reason for revocation.
Compromise Occurrence Date	No	SHALL be specified if the Revocation Reason is 'compromised'.

Formatted Table
Deleted: Required

1147

Table 142: Revoke Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object

Deleted: Required
Formatted Table

1148

Table 143: Revoke Response Payload

1149 **4.20 Destroy**

1150 This request is used to indicate to the server that the key material for the specified Managed Object
 1151 SHALL be destroyed. The meta-data for the key material MAY be retained by the server (e.g., used to
 1152 ensure that an expired or revoked private signing key is no longer available). Special authentication and
 1153 authorization SHOULD be enforced to perform this request (see Usage Guide). Only the object creator or
 1154 an authorized security officer SHOULD be allowed to issue this request. If the Unique Identifier specifies
 1155 a Template object, then the object itself, including all meta-data, SHALL be destroyed.

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

Deleted: Required
Formatted Table

1156 **Table 144: Destroy Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object

Deleted: Required
Formatted Table

1157 **Table 145: Destroy Response Payload**

1158 **4.21 Archive**

1159 This request is used to specify that a Managed Object MAY be archived. The actual time when the object
 1160 is archived, the location of the archive, or level of archive hierarchy is determined by the policies within
 1161 the key management system and is not specified by the client. The request contains the unique identifier
 1162 of the Managed Object. Special authentication and authorization SHOULD be enforced to perform this
 1163 request (see Usage Guide). Only the object creator or an authorized security officer SHOULD be allowed
 1164 to issue this request. This request is only a "hint" to the key management system to possibly archive the
 1165 object.

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

Formatted Table
Deleted: Required

1166 **Table 146: Archive Request Payload**

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object

Deleted: Required
Formatted Table

1167 **Table 147: Archive Response Payload**

1168 **4.22 Recover**

1169 This request is used to obtain access to a Managed Object that has been archived. This request MAY
 1170 require asynchronous polling to obtain the response due to delays caused by retrieving the object from
 1171 the archive. Once the response is received, the object is now on-line, and MAY be obtained (e.g., via a
 1172 Get operation). Special authentication and authorization SHOULD be enforced to perform this request
 1173 (see Usage Guide).

Request Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	No	Determines the object being recovered. If omitted, then the ID Placeholder is substituted by the server.

Deleted: Required
Formatted Table

1174 Table 148: Recover Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object

Deleted: Required
Formatted Table

1175 Table 149: Recover Response Payload

1176 **4.23 Validate**

1177 This requests that the server validate a certificate chain and return information on its validity. Only a
 1178 single certificate chain SHALL be included in each request. Support for this operation at the server is
 1179 OPTIONAL.

Deleted: optional

1180 The request may contain a list of certificate objects, and/or a list of Unique Identifiers that identify
 1181 Managed Certificate objects. Together, the two lists compose a certificate chain to be validated. The
 1182 request MAY also contain a date for which the certificate chain is REQUIRED to be valid.

Deleted: required

1183 The method or policy by which validation is conducted is a decision of the server and is outside of the
 1184 scope of this protocol. Likewise, the order in which the supplied certificate chain is validated and the
 1185 specification of trust anchors used to terminate validation are also controlled by the server.

Request Payload		
Object	<u>REQUIRED</u>	Description
Certificate	No, <u>MAY</u> be repeated	One or more Certificates.
Unique Identifier	No, <u>MAY</u> be repeated	One or more Unique Identifiers of Certificate Objects.
Validity Date	No	A Date-Time object indicating when the certificate chain is valid.

Deleted: Required
Formatted Table
Deleted: May
Deleted: May

1186 Table 150: Validate Request Payload

Response Payload		
Object	<u>REQUIRED</u>	Description
Validity Indicator	Yes	An Enumeration object indicating whether the certificate chain is valid, invalid, or unknown.

Formatted Table
Deleted: Required

1187 Table 151: Validate Response Payload

1188 **4.24 Query**

1189 This request is used by the client to interrogate the server to determine its capabilities and/or protocol
 1190 mechanisms. The *Query* operation SHOULD be invocable by unauthenticated clients to interrogate server
 1191 features and functions. The *Query Function* field in the request SHALL contain one or more of the
 1192 following items:

- 1193 • Query Operations
- 1194 • Query Objects
- 1195 • Query Server Information
- 1196 • Query Application Namespaces

1197 The *Operation* fields in the response contain Operation enumerated values, which SHALL list the
 1198 OPTIONAL operations that the server supports. If the request contains a Query Operations value in the
 1199 Query Function field, then these fields SHALL be returned in the response. The OPTIONAL operations
 1200 are:

Deleted: optional
 Deleted: optional

- 1201 • Validate
- 1202 • Certify
- 1203 • Re-Certify
- 1204 • Notify
- 1205 • Put

1206 The *Object Type* fields in the response contain Object Type enumerated values, which SHALL list the
 1207 object types that the server supports. If the request contains a *Query Objects* value in the Query Function
 1208 field, then these fields SHALL be returned in the response. The object types (any of which are
 1209 OPTIONAL) are:

Deleted: optional

- 1210 • Certificate
- 1211 • Symmetric Key
- 1212 • Public Key
- 1213 • Private Key
- 1214 • Split Key
- 1215 • Template
- 1216 • Secret Data
- 1217 • Opaque Object

1218 The *Server Information* field in the response is a structure containing vendor-specific fields and/or
 1219 substructures. If the request contains a *Query Server Information* value in the Query Function field, then
 1220 this field SHALL be returned in the response.

1221 The Application Namespace fields in the response contain the namespaces that the server SHALL
 1222 generate values for if requested by the client (see Section 3.28). These fields SHALL only be returned in
 1223 the response if the request contains a Query Application Namespaces value in the Query Function field.

1224 Note that the response payload is empty if there are no values to return.

Request Payload		
Object	<u>REQUIRED</u>	Description
Query Function	Yes, <u>MAY</u> be Repeated	Determines the information being queried

Formatted Table
 Deleted: Required
 Deleted: May

1225 **Table 152: Query Request Payload**

Response Payload		
Object	REQUIRED	Description
Operation	No, <u>MAY</u> be repeated	Specifies an Operation that is supported by the server. Only <u>OPTIONAL</u> operations SHALL be listed.
Object Type	No, <u>MAY</u> 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	No, <u>MAY</u> be repeated	Specifies an Application Namespace supported by the server.

- Formatted Table
- Deleted: Required
- Deleted: May
- Deleted: optional
- Deleted: May
- Deleted: May

1226 **Table 153: Query Response Payload**

1227 **4.25 Cancel**

1228 This request is used to cancel an outstanding asynchronous operation. The correlation value (see Section
 1229 6.8) of the original operation SHALL be specified in the request. The server SHALL respond with a
 1230 *Cancellation Result* that contains one of the following values:

- 1231 • *Canceled* – The cancel operation succeeded in canceling the pending operation.
- 1232 • *Unable To Cancel* – The cancel operation is unable to cancel the pending operation.
- 1233 • *Completed* – The pending operation completed successfully before the cancellation operation
 1234 was able to cancel it.
- 1235 • *Failed* – The pending operation completed with a failure before the cancellation operation was
 1236 able to cancel it.
- 1237 • *Unavailable* – The specified correlation value did not match any recently pending or completed
 1238 asynchronous operations.

1239 The response to this operation is not able to be asynchronous.

Request Payload		
Object	REQUIRED	Description
Asynchronous Correlation Value	Yes	Specifies the request being canceled

- Deleted: Required
- Formatted Table

1240 **Table 154: Cancel Request Payload**

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

- Formatted Table
- Deleted: Required

1241 **Table 155: Cancel Response Payload**

1242 **4.26 Poll**

1243 This request is used to poll the server in order to obtain the status of an outstanding asynchronous
 1244 operation. The correlation value (see Section 6.8) of the original operation SHALL be specified in the
 1245 request. The response to this operation is not able to be asynchronous.

Request Payload		
Object	<u>REQUIRED</u>	Description
Asynchronous Correlation Value.	Yes	Specifies the request being polled

Deleted: Required
 Formatted Table

1246 **Table 156: Poll Request Payload**

1247 The server SHALL reply with one of two responses:

1248 If the operation has not completed, the response SHALL contain no payload and a Result Status of
 1249 Pending.

1250 If the operation has completed, the response SHALL contain the appropriate payload for the operation.
 1251 This response SHALL be identical to the response that would have been sent if the operation had
 1252 completed synchronously.

1253 **5 Server-to-Client Operations**

1254 Server-to-client operations are used by servers to send information or Managed Cryptographic Objects to
 1255 clients via means outside of the normal client-server request-response mechanism. These operations are
 1256 used to send Managed Cryptographic Objects directly to clients without a specific request from the client.

1257 **5.1 Notify**

1258 This operation is used to notify a client of events that resulted in changes to attributes of an object. This
 1259 operation is only ever sent by a server to a client via means outside of the normal client request/response
 1260 protocol, using information known to the server via unspecified configuration or administrative
 1261 mechanisms. It contains the Unique Identifier of the object to which the notification applies, and a list of
 1262 the attributes whose changed values have triggered the notification. The message is sent as a normal
 1263 Request message, except that the Maximum Response Size, Asynchronous Indicator, Batch Error
 1264 Continuation Option, and Batch Order Option fields are not allowed. The client SHALL send a response in
 1265 the form of a Response Message containing no payload, unless both the client and server have prior
 1266 knowledge (obtained via out-of-band mechanisms) that the client is not able to respond. Server and Client
 1267 support for this message is OPTIONAL.

Deleted: optional

Message Payload		
Object	<u>REQUIRED</u>	Description
Unique Identifier	Yes	The Unique Identifier of the object.
Attribute	Yes, <u>MAY</u> be repeated	The attributes that have changed. This includes at least the Last Changed Date attribute.

Formatted Table
 Deleted: Required
 Deleted: May

1268 **Table 157: Notify Message Payload**

1269 **5.2 Put**

1270 This operation is used to “push” Managed Cryptographic Objects to clients. This operation is only ever
 1271 sent by a server to a client via means outside of the normal client request/response protocol, using
 1272 information known to the server via unspecified configuration or administrative mechanisms. It contains
 1273 the Unique Identifier of the object that is being sent, and the object itself. The message is sent as a
 1274 normal Request message, except that the Maximum Response Size, Asynchronous Indicator, Batch Error
 1275 Continuation Option, and Batch Order Option fields are not allowed. The client SHALL send a response in

1276 the form of a Response Message containing no payload, unless both the client and server have prior
 1277 knowledge (obtained via out-of-band mechanisms) that the client is not able to respond. Server and client
 1278 support for this message is **OPTIONAL**.

Deleted: optional

1279 The *Put Function* field indicates whether the object being “pushed” is a new object, or is a replacement for
 1280 an object already known to the client (e.g., when pushing a certificate to replace one that is about to
 1281 expire, the Put Function field would be set to indicate replacement, and the Unique Identifier of the
 1282 expiring certificate would be placed in the *Replaced Unique Identifier* field). The Put Function SHALL
 1283 contain one of the following values:

- 1284 • *New* – which indicates that the object is not a replacement for another object.
- 1285 • *Replace* – which indicates that the object is a replacement for another object, and that the
 1286 Replaced Unique Identifier field is present and contains the identification of the replaced object.

1287 The Attribute field contains one or more attributes that the server is sending along with the object. The
 1288 server MAY include attributes with the object to specify how the object is to be used by the client. The
 1289 server MAY include a Lease Time attribute that grants a lease to the client.

1290 If the Managed Object is a wrapped key, then the key wrapping specification SHALL be exchanged prior
 1291 to the transfer via out-of-band mechanisms.

Message Payload		
Object	REQUIRED	Description
Unique Identifier	Yes	The Unique Identifier of the object.
Put Function	Yes	Indicates function for Put message.
Replaced Unique Identifier	No	Unique Identifier of 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	Yes	The object being sent to the client.
Attribute	No, MAY be repeated	The additional attributes that the server wishes to send with the object.

Formatted Table

Deleted: Required

Deleted: May

1292 **Table 158: Put Message Payload**

1293 6 Message Contents

1294 The messages in the protocol consist of a message header, one or more batch items (which contain
 1295 **OPTIONAL** message payloads), and **OPTIONAL** message extensions. The message headers contain
 1296 fields whose presence is determined by the protocol features used (e.g., asynchronous responses). The
 1297 field contents are also determined by whether the message is a request or a response. The message
 1298 payload is determined by the specific operation being requested or to which is being replied.

Deleted: optional

Deleted: optional

1299 The message headers are structures that contain some of the following objects.

1300 6.1 Protocol Version

1301 This field contains the version number of the protocol, ensuring that the protocol is fully understood by
 1302 both communicating parties. The version number is specified in two parts, major and minor. Servers and
 1303 clients SHALL support backward compatibility with versions of the protocol with the same major version.
 1304 Support for backward compatibility with different major versions is **OPTIONAL**.

Deleted: optional

Object	Encoding	<u>REQUIRED</u>
Protocol Version	Structure	Yes
Protocol Version Major	Integer	Yes
Protocol Version Minor	Integer	Yes

Deleted: Required

1305 **Table 159: Protocol Version Structure in Message Header**

1306 6.2 Operation

1307 This field indicates the operation being requested or the operation for which the response is being
1308 returned. The operations are defined in Sections 4 and 5 .

Object	Encoding	<u>REQUIRED</u>
Operation	Enumeration	Yes

Deleted: Required

1309 **Table 160: Operation in Batch Item**

1310 6.3 Maximum Response Size

1311 This field is optionally contained in a request message, and is used to indicate the maximum size of a
1312 response that the requester SHALL handle. It SHOULD only be sent in requests that possibly return large
1313 replies.

Object	Encoding	<u>REQUIRED</u>
Maximum Response Size	Integer	No

Deleted: Required

1314 **Table 161: Maximum Response Size in Message Request Header**

1315 6.4 Unique Batch Item ID

1316 This field is optionally contained in a request, and is used for correlation between requests and
1317 responses. If a request has a *Unique Batch Item ID*, then responses to that request SHALL have the
1318 same Unique Batch Item ID.

Object	Encoding	<u>REQUIRED</u>
Unique Batch Item ID	Octet String	No

Deleted: Required

1319 **Table 162: Unique Batch Item ID in Batch Item**

1320 6.5 Time Stamp

1321 This field is optionally contained in a request, is REQUIRED in a response, is used for time stamping, and
1322 MAY be used to enforce reasonable time usage at a client (e.g., a server MAY choose to reject a request
1323 if a client's time stamp contains a value that is too far off the known correct time). Note: the time stamp
1324 MAY be used by a client that has no real-time clock but has a countdown timer, to obtain useful "seconds
1325 from now" values from all of the Date attributes by performing a subtraction.

Deleted: required

Object	Encoding	<u>REQUIRED</u>
Time Stamp	Date-Time	No

Deleted: Required

1326 **Table 163: Time Stamp in Message Header**

1327 6.6 Authentication

1328 | This is used to authenticate the requester. It is an OPTIONAL information item, depending on the type of
1329 | request being issued and on server policies. Servers MAY require authentication on no requests, a
1330 | subset of the requests, or all requests, depending on policy. Query operations used to interrogate server
1331 | features and functions SHOULD NOT require authentication.

Deleted: optional

1332 | The authentication mechanisms are described and discussed in Section 8 .

Object	Encoding	<u>REQUIRED</u>
Authentication	Structure	No
Credential	Structure	Yes

Deleted: Required

1333 | **Table 164: Authentication Structure in Message Header**

1334 | The Credential structure is defined in Section 2.1.2 .

1335 6.7 Asynchronous Indicator

1336 | This Boolean flag indicates whether the client is able to accept an asynchronous response. It SHALL
1337 | have the Boolean value True if the client is able to handle asynchronous responses, and the value False
1338 | otherwise. If not present in a request, then False is assumed. If a client indicates that it is not able to
1339 | handle asynchronous responses (i.e., flag is set to False), and the server is not able to process the
1340 | request synchronously, then the server SHALL respond to the request with a failure.

Object	Encoding	<u>REQUIRED</u>
Asynchronous Indicator	Boolean	No

Deleted: Required

1341 | **Table 165: Asynchronous Indicator in Message Request Header**

1342 6.8 Asynchronous Correlation Value

1343 | This is returned in the immediate response to an operation that requires asynchronous polling. Note: the
1344 | server decides which operations are performed synchronously or asynchronously. A server-generated
1345 | correlation value SHALL be specified in any subsequent Poll or Cancel operations that pertain to the
1346 | original operation.

Object	Encoding	<u>REQUIRED</u>
Asynchronous Correlation Value	Octet String	No

Deleted: Required

1347 | **Table 166: Asynchronous Correlation Value in Response Batch Item**

1348 6.9 Result Status

1349 | This is sent in a response message and indicates the success or failure of a request. The following values
1350 | MAY be set in this field:

- 1351 | • *Success* – The requested operation completed successfully.
- 1352 | • *Pending* – The requested operation is in progress, and it is necessary to obtain the actual result
1353 | via asynchronous polling. The asynchronous correlation value SHALL be used for the subsequent
1354 | polling of the result status.
- 1355 | • *Undone* – The requested operation was performed, but had to be undone (i.e., due to a failure in
1356 | a batch for which the Error Continuation Option was set to Undo).
- 1357 | • *Failure* – The requested operation failed.

Object	Encoding	<u>REQUIRED</u>
Result Status	Enumeration	Yes

Deleted: Required

1358

Table 167: Result Status in Response Batch Item

1359 6.10 Result Reason

1360 This field indicates a reason for failure or a modifier for a partially successful operation and SHALL be
 1361 present in responses that return a Result Status of Failure. It is OPTIONAL in any response that returns a
 1362 Result Status of Success. The following defined values MAY be set in this field:

Deleted: optional

- 1363 • *Item not found* – A requested object was not found or did not exist.
- 1364 • *Response too large* – The response to a request would exceed the *Maximum Response Size* in
 1365 the request.
- 1366 • *Authentication not successful* – The authentication information in the request was not able to be
 1367 validated, or there was no authentication information in the request when there SHOULD have
 1368 been.
- 1369 • *Invalid message* – The request message was not understood by the server.
- 1370 • *Operation not supported* – The operation requested by the request message is not supported by
 1371 the server.
- 1372 • *Missing data* – The operation requires additional OPTIONAL information in the request, which
 1373 was not present.
- 1374 • *Invalid field* – Some data item in the request has an invalid value.
- 1375 • *Feature not supported* – An OPTIONAL feature specified in the request is not supported.
- 1376 • *Operation canceled by requester* – The operation was asynchronous, and the operation was
 1377 canceled by the Cancel operation before it completed successfully.
- 1378 • *Cryptographic failure* – The operation failed due to a cryptographic error.
- 1379 • *Illegal operation* – The client requested an operation that was not able to be performed with the
 1380 specified parameters.
- 1381 • *Permission denied* – The client does not have permission to perform the requested operation.
- 1382 • *Object archived* – The object SHALL be recovered from the archive before performing the
 1383 operation.
- 1384 • *General failure* – The request failed for a reason other than the defined reasons above.

Deleted: optional

Deleted: optional

Object	Encoding	<u>REQUIRED</u>
Result Reason	Enumeration	Yes

Deleted: Required

1385

Table 168: Result Reason in Response Batch Item

1386 6.11 Result Message

1387 This field MAY be returned in a response. It contains a more descriptive error message, which MAY be
 1388 used by the client to display to an end user or for logging/auditing purposes.

Object	Encoding	<u>REQUIRED</u>
Result Message	Text String	No

Deleted: Required

1389

Table 169: Result Message in Response Batch Item

1390 **6.12 Batch Order Option**

1391 A Boolean value used in requests where the Batch Count is greater than 1. If True, then batched
 1392 operations SHALL be executed in the order in which they appear within the request. If False, then the
 1393 server MAY choose to execute the batched operations in any order. If not specified, then False is
 1394 assumed (i.e., no implied ordering). Server support for this feature is **OPTIONAL**, but if the server does
 1395 not support the feature, and a request is received with the batch order option set to True, then the entire
 1396 request SHALL be rejected.

Deleted: optional

Object	Encoding	REQUIRED
Batch Order Option	Boolean	No

Deleted: Required

1397 **Table 170: Batch Order Option in Message Request Header**

1398 **6.13 Batch Error Continuation Option**

1399 This option SHALL only be present if the Batch Count is greater than 1. This option SHALL have one of
 1400 three values:

- 1401 • *Undo* – If any operation in the request fails, then the server SHALL undo all the previous
 1402 operations.
- 1403 • *Stop* – If an operation fails, then the server SHALL NOT continue processing subsequent
 1404 operations in the request. Completed operations SHALL NOT be undone.
- 1405 • *Continue* – Return an error for the failed operation, and continue processing subsequent
 1406 operations in the request.

1407 If not specified, then Stop is assumed.

1408 Server support for this feature is **OPTIONAL**, but if the server does not support the feature, and a request
 1409 is received containing the *Batch Error Continuation* option with a value other than the default Stop, then
 1410 the entire request SHALL be rejected.

Deleted: optional

Object	Encoding	REQUIRED
Batch Error Continuation Option	Enumeration	No

Deleted: Required

1411 **Table 171: Batch Error Continuation Option in Message Request Header**

1412 **6.14 Batch Count**

1413 This field contains the number of Batch Items in a message and is **REQUIRED**. If only a single operation
 1414 is being requested, then the batch count SHALL be set to 1. The Message Payload, which follows the
 1415 Message Header, contains one or more batch items.

Deleted: required

Object	Encoding	REQUIRED
Batch Count	Integer	Yes

Deleted: Required

1416 **Table 172: Batch Count in Message Header**

1417 **6.15 Batch Item**

1418 This field consists of a structure that holds the individual requests or responses in a batch, and is
 1419 **REQUIRED**. The contents of the batch items are described in Sections 7.2 and 7.3.

Deleted: required

Object	Encoding	REQUIRED
Batch Item	Structure	No

Deleted: Required

1420

Table 173: Batch Item in Message

1421 6.16 Message Extension

1422 | The *Message Extension* is an **OPTIONAL** structure that MAY be appended to any Batch Item. It is used
 1423 | to extend protocol messages for the purpose of adding vendor specified extensions. The Message
 1424 | Extension is a structure containing a Vendor Identification, a Criticality Indicator, and vendor-specific
 1425 | extensions. The *Vendor Identification* SHALL be a text string that uniquely identifies the vendor, allowing
 1426 | a client to determine if it is able to parse and understand the extension. If a client or server receives a
 1427 | protocol message containing a message extension that it does not understand, then its actions depend
 1428 | on the *Criticality Indicator*. If the indicator is True (i.e., Critical), and the receiver does not understand the
 1429 | extension, then the receiver SHALL reject the entire message. If the indicator is False (i.e., Non-Critical),
 1430 | and the receiver does not understand the extension, then the receiver MAY process the rest of the
 1431 | message as if the extension were not present.

Deleted: optional

Object	Encoding	REQUIRED
Message Extension	Structure	No
Vendor Identification	Text String	Yes
Criticality Indicator	Boolean	Yes
Vendor Extension	Structure	Yes

Deleted: Required

1432

Table 174: Message Extension Structure in Batch Item

1433 7 Message Format

1434 Messages contain the following objects and fields. All fields SHALL appear in the order specified.

1435 7.1 Message Structure

Object	Encoding	REQUIRED
Request Message	Structure	Yes
Request Header	Structure	Yes
Batch Item	Structure	Yes, MAY be repeated

Deleted: Required

Deleted: May

1436

Table 175: Request Message Structure

Object	Encoding	REQUIRED
Response Message	Structure	Yes
Response Header	Structure	Yes
Batch Item	Structure	Yes, MAY be repeated

Deleted: Required

Deleted: May

1437

Table 176: Response Message Structure

Synchronous Request Header		
Object	<u>REQUIRED</u> in Message	Comment
Request Header	Yes	Structure
Protocol Version	Yes	
Maximum Response Size	No	
Authentication	No	
Batch Error Continuation Option	No	If omitted, then Stop is assumed
Batch Order Option	No	If omitted, then False is assumed
Time Stamp	No	
Batch Count	Yes	

Deleted: Required

1439 Table 177: Synchronous Request Header Structure

Synchronous Request Batch Item		
Object	<u>REQUIRED</u> in Message	Comment
Batch Item	Yes	Structure
Operation	Yes	
Unique Batch Item ID	No	<u>REQUIRED</u> if <i>Batch Count</i> > 1
Request Payload	Yes	Structure, contents depend on the Operation
Message Extension	No	

Deleted: Required

Deleted: Required

1440 Table 178: Synchronous Request Batch Item Structure

Synchronous Response Header		
Object	<u>REQUIRED</u> in Message	Comment
Response Header	Yes	Structure
Protocol Version	Yes	
Time Stamp	Yes	
Batch Count	Yes	

Deleted: Required

1441 Table 179: Synchronous Response Header Structure

Synchronous Response Batch Item		
Object	<u>REQUIRED</u> in Message	Comment
Batch Item	Yes	Structure
Operation	Yes, if not a failure	
Unique Batch Item ID	No	<u>REQUIRED</u> if <i>Batch Count</i> > 1
Result Status	Yes	
Result Reason	No	Only present if Result Status is not <i>Success</i>
Result Message	No	Only present if Result Status is not <i>Success</i>
Response Payload	Yes, if not a failure	Structure, contents depend on the Operation
Message Extension	No	

Deleted: Required

Deleted: Required

1442

Table 180: Synchronous Response Batch Item Structure

1443 7.3 Asynchronous Operations

1444 If the client is capable of accepting asynchronous responses, then it MAY set the *Asynchronous Indicator*
 1445 in the header of a batched request. The batched responses MAY contain a mixture of synchronous and
 1446 asynchronous responses.

Asynchronous Request Header		
Object	<u>REQUIRED</u> in Message	Comment
Request Header	Yes	Structure
Protocol Version	Yes	
Maximum Response Size	No	
Asynchronous Indicator	Yes	SHALL be set to True
Authentication	No	
Batch Error Continuation Option	No	If omitted, then Stop is assumed
Batch Order Option	No	If omitted, then False is assumed
Time Stamp	No	
Batch Count	Yes	

Deleted: Required

1447

Table 181: Asynchronous Request Header Structure

Asynchronous Request Batch Item		
Object	<u>REQUIRED</u> in Message	Comment
Batch Item	Yes	Structure
Operation	Yes	
Unique Batch Item ID	No	<u>REQUIRED</u> if <i>Batch Count</i> > 1
Request Payload	Yes	Structure, contents depend on the Operation
Message Extension	No	

Deleted: Required

Deleted: Required

Table 182: Asynchronous Request Batch Item Structure

Asynchronous Response Header		
Object	<u>REQUIRED</u> in Message	Comment
Response Header	Yes	Structure
Protocol Version	Yes	
Time Stamp	Yes	
Batch Count	Yes	

Deleted: Required

Table 183: Asynchronous Response Header Structure

Asynchronous Response Batch Item		
Object	<u>REQUIRED</u> in Message	Comment
Batch Item	Yes	Structure
Operation	Yes, if not a failure	
Unique Batch Item ID	No	<u>REQUIRED</u> if <i>Batch Count</i> > 1
Result Status	Yes	
Result Reason	No	Only present if Result Status is not <i>Pending</i> or <i>Success</i>
Result Message	No	Only present if Result Status is not <i>Pending</i> or <i>Success</i>
Asynchronous Correlation Value	Yes	Only present if Result Status is <i>Pending</i>
Response Payload	Yes, if not a failure	Structure, contents depend on the Operation
Message Extension	No	

Deleted: Required

Deleted: Required

Table 184: Asynchronous Response Batch Item Structure

8 Authentication

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

1456 The Usage Guide describes authentication profiles appropriate to this protocol, as well as the relationship
1457 of those mechanisms to the credentials that are optionally included in the Authentication field. The
1458 authentication profiles described are:

- 1459 • SSL/TLS authentication. If the transport protocol uses a normal TCP stream, then that stream
1460 SHOULD use an SSL/TLS encryption layer, and the client and server authentication features
1461 SHALL be enabled unless otherwise specified in the operation. The Credential object contained
1462 in the Authentication field in all request messages SHALL contain the client's certificate. The
1463 server SHOULD use this certificate to identify the client for policy enforcement purposes, and
1464 SHOULD verify that this certificate matches the one used for SSL/TLS authentication.
- 1465 • HTTPS authentication. If the transport protocol is HTTP over TCP, then the HTTPS protocol
1466 SHOULD be used, and the client and server authentication features enabled unless otherwise
1467 specified in the operation. The contents and use of the *Credential* object are the same as in the
1468 case of SSL/TLS above.

1469 All server implementations SHOULD, at a minimum, support the SSL/TLS and HTTPS profiles described
1470 in the Usage Guide.

1471 Other mechanisms (e.g., Kerberos) are potentially usable, with the identity established in the mechanism
1472 (e.g., the Kerberos token), expressed as the Credential object. Profiles for these mechanisms are not
1473 currently described in the Usage Guide.

1474 9 Message Encoding

1475 To support different transport protocols and different client capabilities, a number of message-encoding
1476 mechanisms are supported.

1477 9.1 TTLV Encoding

1478 In order to minimize the resource impact on potentially low-function clients, one encoding mechanism to
1479 be used for protocol messages is a simplified TTLV (Tag, Type, Length, Value) scheme.

1480 The scheme is designed to minimize the CPU cycle and memory requirements of clients that need to
1481 encode or decode protocol messages, and to provide optimal alignment for both 32-bit and 64-bit
1482 processors. Minimizing bandwidth over the transport mechanism is considered to be of lesser importance.

1483 9.1.1 TTLV Encoding Fields

1484 Every Data object encoded by the TTLV scheme consists of 4 items, in order:

1485 9.1.1.1 Item Tag

1486 An Item Tag is a 3-byte binary unsigned integer, transmitted big endian, which contains a number that
1487 designates the specific Protocol Field or Object that the TTLV object represents. To ease debugging, and
1488 to ensure that malformed messages are detected more easily, all tags SHALL contain either the value 42
1489 in hex or the value 54 in hex as the high order (first) byte. Tags defined by this specification contain hex
1490 42 in the first byte. Extensions, which are permitted, but are not defined in this specification, contain the
1491 value 54 hex in the first byte. A list of defined Item Tags is in Section 9.1.3.1 .

1492 9.1.1.2 Item Type

1493 An Item Type is a byte containing a coded value that indicates the data type of the data object. The
1494 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
Octet String	08
Date-Time	09
Interval	0A

Table 185: Allowed Item Type Values

1495

1496 **9.1.1.3 Item Length**

1497 An Item Length is a 32-bit binary integer, transmitted big-endian, containing the number of bytes in the
1498 Item Value. The allowed values are:

1499

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
Octet String	Varies
Date-Time	8
Interval	4

Table 186: Allowed Item Length Values

1500

1501 If the Item Type is Structure, then the Item Length is the total length of all of the sub-items contained in
1502 the structure, including any padding. If the Item Type is Integer, Enumeration, Text String, Octet String, or

1503 Interval, then the Item Length is the number of bytes excluding the padding bytes. Text Strings and Octet
1504 Strings SHALL be padded with the minimal number of bytes following the Item Value to obtain a multiple
1505 of 8 bytes. Integers, Enumerations, and Intervals SHALL be padded with 4 bytes following the Item Value.

1506 9.1.1.4 Item Value

1507 The item value is a sequence of bytes containing the value of the data item, depending on the type:

- 1508 • Integers are encoded as 4-byte long (32 bit) binary signed numbers in 2's complement notation,
1509 transmitted big-endian.
- 1510 • Long Integers are encoded as 8-byte long (64 bit) binary signed numbers in 2's complement
1511 notation, transmitted big-endian.
- 1512 • Big Integers are encoded as a sequence of 8-bit bytes, in 2's complement notation, transmitted
1513 big-endian. If the length of the sequence is not a multiple of 8 bytes, then Big Integers SHALL be
1514 padded with the minimal number of leading sign-extended bytes to make the length a multiple of
1515 8 bytes. These padding bytes are part of the Item Value and SHALL be counted in the Item
1516 Length.
- 1517 • Enumerations are encoded as 4-byte long (32 bit) binary unsigned numbers transmitted big-
1518 endian. Extensions, which are permitted, but are not defined in this specification, contain the
1519 value 8 hex in the first nibble of the first byte.
- 1520 • Booleans are encoded as an 8-byte value that SHALL either contain the hex value
1521 0000000000000000, indicating the Boolean value *False*, or the hex value 0000000000000001,
1522 transmitted big-endian, indicating the Boolean value *True*.
- 1523 • Text Strings are sequences of bytes encoding character values according to the UTF-8 encoding
1524 standard. There SHALL be no null-termination at the end of such strings.
- 1525 • Octet Strings are sequences of bytes containing individual unspecified 8 bit binary values.
- 1526 • Date-Time values are encoded as 8-byte long (64 bit) binary signed numbers, transmitted big-
1527 endian. They are POSIX Time values (described in IEEE Standard 1003.1) extended to a 64 bit
1528 value to eliminate the "Year 2038 problem" (i.e., problem that affects Unix systems that store time
1529 as a signed 32-bit integer). The value is expressed as the number of seconds from a time epoch,
1530 which is 00:00:00 GMT January 1st, 1970. This value has a resolution of 1 second. All Date-Time
1531 values are expressed as UTC values.
- 1532 • Intervals are encoded as 4-byte long (32 bit) binary unsigned numbers, transmitted big-endian.
1533 They have a resolution of 1 second.
- 1534 • Structure Values are encoded as the concatenated encodings of the elements of the structure. All
1535 structures defined in this specification SHALL have all of their fields encoded in the order in which
1536 they appear in their respective structure descriptions.

1537 9.1.2 Examples

1538 These examples are assumed to be encoding a Protocol Object whose tag is 420020. The examples are
1539 shown as a sequence of bytes in hexadecimal notation:

- 1540 • An Integer containing the decimal value 8:
1541 42 00 20 | 02 | 00 00 00 04 | 00 00 00 08 00 00 00 00
- 1542 • A Long Integer containing the decimal value 123456789000000000:
1543 42 00 20 | 03 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00
- 1544 • A Big Integer containing the decimal value 1234567890000000000000000000:
1545 42 00 20 | 04 | 00 00 00 10 | 00 00 00 00 03 FD 35 EB 6B C2 DF 46 18 08
1546 00 00
- 1547 • An Enumeration with value 255:

1548 42 00 20 | 05 | 00 00 00 04 | 00 00 00 FF 00 00 00 00

1549 • A Boolean with the value *True*:

1550 42 00 20 | 06 | 00 00 00 08 | 00 00 00 00 00 00 00 01

1551 • A Text String:

1552 42 00 20 | 07 | 00 00 00 0B | 48 65 6C 6C 6F 20 57 6F 72 6C 64 00 00 00

1553 00 00

1554 • An Octet String:

1555 42 00 20 | 08 | 00 00 00 03 | 01 02 03 00 00 00 00 00

1556 • A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:

1557 42 00 20 | 09 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8

1558 • An Interval, containing the value for 10 days:

1559 42 00 20 | 0A | 00 00 00 04 | 00 0D 2F 00 00 00 00 00

1560 • A Structure containing an Enumeration, value 254, followed by an Integer, value 255, having tags

1561 420004 and 420005 respectively:

1562 42 00 20 | 01 | 00 00 00 20 | 42 00 04 | 05 | 00 00 00 04 | 00 00 00 FE

1563 00 00 00 00 | 42 00 05 | 02 | 00 00 00 04 | 00 00 00 FF 00 00 00 00

1564 **9.1.3 Defined Values**

1565 This section specifies the values that are defined by this specification. In all cases where an extension
 1566 mechanism is allowed, this extension mechanism is only able to be used for communication between
 1567 parties that have pre-agreed understanding of the specific extensions.

1568 **9.1.3.1 Tags**

1569 The following table defines the tag values for the objects and primitive data values for the protocol
 1570 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

Tag	
Object	Tag Value
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 Issuer	420014
Certificate Request	420015
Certificate Request Type	420016
Certificate Subject	420017
Certificate Subject Alternative Name	420018
Certificate Subject Distinguished Name	420019
Certificate Type	42001A
Certificate Value	42001B
Common Template-Attribute	42001C
Compromise Date	42001D
Compromise Occurrence Date	42001E
Contact Information	42001F
Credential	420020
Credential Type	420021
Credential Value	420022
Criticality Indicator	420023
CRT Coefficient	420024
Cryptographic Algorithm	420025
Cryptographic Length	420026
Cryptographic Parameters	420027
Cryptographic Usage Mask	420028
Custom Attribute	420029
D	42002A
Deactivation Date	42002B
Derivation Data	42002C
Derivation Method	42002D
Derivation Parameters	42002E

Tag	
Object	Tag Value
Destroy Date	42002F
Digest	420030
Digest Value	420031
Encryption Key Information	420032
G	420033
Hashing Algorithm	420034
Initial Date	420035
Initialization Vector	420036
Issuer	420037
Iteration Count	420038
IV/Counter/Nonce	420039
J	42003A
Key	42003B
Key Block	42003C
Key Material	42003D
Key Part Identifier	42003E
Key Value	42003F
Key Value Type	420040
Key Wrapping Data	420041
Key Wrapping Specification	420042
Last Changed Date	420043
Lease Time	420044
Link	420045
Link Type	420046
Linked Object Identifier	420047
MAC/Signature	420048
MAC/Signature Key Information	420049
Maximum Items	42004A
Maximum Response Size	42004B
Message Extension	42004C
Modulus	42004D
Name	42004E
Name Type	42004F
Name Value	420050
Object Group	420051

Tag	
Object	Tag Value
Object Type	420052
Offset	420053
Opaque Data Type	420054
Opaque Data Value	420055
Opaque Object	420056
Operation	420057
Operation Policy Name	420058
P	420059
Padding Method	42005A
Prime Exponent P	42005B
Prime Exponent Q	42005C
Prime Field Size	42005D
Private Exponent	42005E
Private Key	42005F
Private Key Template-Attribute	420060
Private Key Unique Identifier	420061
Process Start Date	420062
Protect Stop Date	420063
Protocol Version	420064
Protocol Version Major	420065
Protocol Version Minor	420066
Public Exponent	420067
Public Key	420068
Public Key Template-Attribute	420069
Public Key Unique Identifier	42006A
Put Function	42006B
Q	42006C
Q String	42006D
Query Function	42006E
Recommended Curve	42006F
Replaced Unique Identifier	420070
Request Header	420071
Request Message	420072
Request Payload	420073
Response Header	420074
Response Message	420075

Tag	
Object	Tag Value
Response Payload	420076
Result Message	420077
Result Reason	420078
Result Status	420079
Revocation Message	42007A
Revocation Reason	42007B
Revocation Reason Code	42007C
Role Type	42007D
Salt	42007E
Secret Data	42007F
Secret Data Type	420080
Serial Number	420081
Server Information	420082
Split Key	420083
Split Key Method	420084
Split Key Parts	420085
Split Key Threshold	420086
State	420087
Storage Status Mask	420088
Symmetric Key	420089
Template	42008A
Template-Attribute	42008B
Time Stamp	42008C
Unique Batch Item ID	42008D
Unique Identifier	42008E
Usage Limits	42008F
Usage Limits Byte Count	420090
Usage Limits Object Count	420091
Usage Limits Total Bytes	420092
Usage Limits Total Objects	420093
Validity Date	420094
Validity Indicator	420095
Vendor Extension	420096
Vendor Identification	420097
Wrapping Method	420098
X	420099

Deleted: Identifier

Deleted: Batch Item ID

Tag	
Object	Tag Value
Y	42009A
(Reserved)	42009B - 42FFFF
(Unused)	430000 - 53FFFF
Extensions	540000 - 54FFFF
(Unused)	550000 - FFFFFFFF

Table 187: Tag Values

1571

1572 **9.1.3.2 Enumerations**

1573 The following tables define the values for enumerated lists.

1574 **9.1.3.2.1 Credential Type Enumeration**

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

1575 **Table 188: Credential Type Enumeration**

1576 **9.1.3.2.2 Key Value Type Enumeration**

Key Value Type	
Name	Value
Raw	00000001
Opaque	00000002
PKCS#1	00000003
PKCS#8	00000004
X.509	00000005
Transparent Symmetric Key	00000006
Transparent DSA Private Key	00000007
Transparent DSA Public Key	00000008
Transparent RSA Private Key	00000009
Transparent RSA Public Key	0000000A
Transparent DH Private Key	0000000B
Transparent DH Public Key	0000000C
Transparent ECDSA Private Key	0000000D
Transparent ECDSA Public Key	0000000E
Transparent ECDH Private Key	0000000F
Transparent ECDH Public Key	00000010
Extensions	8XXXXXXXX

1577 **Table 189: Key Value Type Enumeration**

1578 **9.1.3.2.3 Wrapping Method Enumeration**

Wrapping Method	
Name	Value
Encrypt	00000001
MAC/sign	00000002
Encrypt then MAC/sign	00000003
MAC/sign then encrypt	00000004
TR-31	00000005
Extensions	8XXXXXXXX

1579 **Table 190: Wrapping Method Enumeration**

1580 **9.1.3.2.4 Recommended Curve Enumeration for ECDSA and ECDH**

1581 Recommended curves are defined in NIST FIPS PUB 186-3.

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

1582 **Table 191: Recommended Curve Enumeration for ECDSA and ECDH**

1583 **9.1.3.2.5 Certificate Type Enumeration**

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

1584 **Table 192: Certificate Type Enumeration**

1585 **9.1.3.2.6 Split Key Method Enumeration**

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

1586 **Table 193: Split Key Method Enumeration**

1587 **9.1.3.2.7 Secret Data Type Enumeration**

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

1588 **Table 194: Secret Data Type Enumeration**

1589 **9.1.3.2.8 Opaque Data Type Enumeration**

Opaque Data Type	
Name	Value
Extensions	8XXXXXXXX

1590 **Table 195: Opaque Data Type Enumeration**

1591 **9.1.3.2.9 Name Type Enumeration**

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

1592 **Table 196: Name Type Enumeration**

1593 **9.1.3.2.10 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

1594 **Table 197: Object Type Enumeration**

1595 **9.1.3.2.11 Cryptographic Algorithm Enumeration**

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

1596 **Table 198: Cryptographic Algorithm Enumeration**

1597 **9.1.3.2.12 Block Cipher Mode Enumeration**

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

1598 **Table 199: Block Cipher Mode Enumeration**

1599 **9.1.3.2.13 Padding Method Enumeration**

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

1600 **Table 200: Padding Method Enumeration**

1601 **9.1.3.2.14 Hashing Algorithm Enumeration**

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

1602 **Table 201: Hashing Algorithm Enumeration**

1603 **9.1.3.2.15 Role Type Enumeration**

Role Type	
Name	Value
BDK	00000001
CVK	00000002
DEK	00000003
MKAC	00000004
MKSMC	00000005
MKSMI	00000006
MKDAC	00000007
MKDN	00000008
MKCP	00000009
KMOTH	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 202: Role Type Enumeration

1604
 1605 Note that while the set and definitions of role types are chosen to match TR-31 there is no necessity to
 1606 match binary representations.

1607 **9.1.3.2.16 State Enumeration**

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

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

1608

Table 203: State Enumeration

1609 **9.1.3.2.17 Revocation Reason Code Enumeration**

Revocation Reason Code	
Name	Value
Key Compromise	00000001
CA Compromise	00000002
Affiliation Changed	00000003
Superseded	00000004
Cessation of Operation	00000005
Certificate Hold	00000006
Privilege Withdrawn	00000007
Revoked By creator	00000008
Revoked By Administrator	00000009
Extensions	8XXXXXXXX

1610

Table 204: Revocation Reason Code Enumeration

1611 **9.1.3.2.18 Link Type Enumeration**

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

1612

Table 205: Link Type Enumeration

1613

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

1614 **9.1.3.2.19 Derivation Method Enumeration**

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

1615 **Table 206: Derivation Method Enumeration**

1616 **9.1.3.2.20 Certificate Request Type Enumeration**

Certificate Request Type	
Name	Value
PCKS#10	00000001
PEM	00000002
PGP	00000003
Extensions	8XXXXXXXX

1617 **Table 207: Certificate Request Type Enumeration**

1618 **9.1.3.2.21 Validity Indicator Enumeration**

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

1619 **Table 208: Validity Indicator Enumeration**

1620 **9.1.3.2.22 Query Function Enumeration**

Query Function	
Name	Value
Query Operations	00000001
Query Objects	00000002
Query Server Information	00000003
Query Application Namespaces	00000004

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

1621

Table 209: Query Function Enumeration

1622 **9.1.3.2.23 Cancellation Result Enumeration**

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

1623

Table 210: Cancellation Result Enumeration

1624 **9.1.3.2.24 Put Function Enumeration**

Put Function	
Name	Value
New	00000001
Replace	00000002
Extensions	8XXXXXXXX

1625

Table 211: Put Function Enumeration

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

Table 212: Operation Enumeration

1628 **9.1.3.2.26 Result Status Enumeration**

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

1629 **Table 213: Result Status Enumeration**

1630 **9.1.3.2.27 Result Reason Enumeration**

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

1631 **Table 214: Result Reason Enumeration**

1632 **9.1.3.2.28 Batch Error Continuation Enumeration**

Batch Error Continuation	
Name	Value
Continue	00000001
Stop	00000002
Undo	00000003

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

1633

Table 215: Batch Error Continuation Enumeration

1634 **9.1.3.3 Bit Masks**

1635 **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

Deleted: x

1636

Table 216: Cryptographic Usage Mask

1637 This list takes into consideration values which MAY appear in the Key Usage extension in an X.509
 1638 certificate.

1639 **9.1.3.3.2 Storage Status Mask**

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

1640 **Table 217: Storage Status Mask**

1641 **9.2 XML Encoding**

1642 An XML Encoding has not yet been defined.

1643 **10 Transport**

1644 This section describes two KMIP transport profiles. These profiles describe mechanisms by which
1645 authentication and communications privacy are established outside KMIP.

1646 **10.1 SSL/TLS Profile**

1647 Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic
1648 protocols that provide secure communications for data transfers, using cryptographic mechanisms to
1649 provide both the authentication of participants and privacy of the communication. SSL 2.0 has known
1650 security issues, and all current implementations of HTTP/S support more recent protocols. Therefore, this
1651 profile prohibits the use of SSL 2.0 and requires SSL 3.1 or TLS 1.0 or later.

1652 In this profile, a KMIP client and server SHALL use SSL/TLS to negotiate a mutually-authenticated
1653 connection. This is REQUIRED for all KMIP communication except for the Query operation.

1654 In SSL and TLS, choices of algorithms are expressed as cipher suites. The following subsections specify
1655 cipher suites that are either REQUIRED or discouraged. The use of any other cipher suite not discussed
1656 below is OPTIONAL.

1657 **10.1.1 Mandatory cipher suites**

1658 The mandatory cipher suites for the SSL/TLS profile are:

- 1659 • A TLS-capable instance SHALL support TLS_RSA_WITH_AES_128_CBC_SHA
- 1660 • An SSL-capable instance SHALL support SSL_RSA_WITH_AES_128_CBC_SHA

1661 **10.1.2 Discouraged cipher suites**

1662 As discussed in "WS-I Basic Security Profile", the cipher suites defined in the SSL and TLS specifications
1663 that use anonymous Diffie-Hellman (i. e. those that have *DH anon* in their symbolic name) are vulnerable
1664 to man-in-the-middle attacks. Such cipher suites SHALL be avoided. This profile disallows the use of the
1665 following cipher suites due to their lack of confidentiality services:

- 1666 • SSL_RSA_WITH_NULL_SHA
- 1667 • TLS_RSA_WITH_NULL_SHA
- 1668 • SSL_RSA_WITH_NULL_MD5
- 1669 • TLS_RSA_WITH_NULL_MD5

1670 Also, cipher suites that use 40 or 56 bit keys SHALL be avoided, due to their relative ease of compromise
1671 through brute-force attack.

Deleted: Transport protocols are not part of the message definitions, and are external to this protocol. The Usage Guide, however, describes two profiles for implementation of this protocol over secure transport protocols, namely:
 <#>SSL/TLS over TCP. This profile describes the implementation of this protocol using SSL/TLS encryption, with client and server authentication features enabled, over a normal TCP stream.
 <#>HTTPS over TCP. This profile describes the implementation of this protocol using HTTPS, with client and server authentication features enabled, over a normal TCP stream.
 To ensure a base level of interoperability, all server implementations SHOULD, at least, support the SSL/TLS and HTTPS transport protocols as described in the Usage Guide.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

1672
1673
1674
1675
1676
1677
1678

1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707

10.2 HTTPS Profile

Hypertext Transfer Protocol over Secure Socket Layer or https is a URI (Universal Resource Indicator) scheme that is used to indicate a secure HTTP connection, requiring an additional encryption and authentication layer, implemented by SSL/TLS, between HTTP and TCP. The establishment of the trust relationship between the client and server is the same as in the SSL/TLS profile described above.

As in the SSL/TLS profile, mutual authentication SHALL be performed for all KMIP communication unless otherwise specified in the operation.

Formatted: Bullets and Numbering

Comment: Sentence "a different default TCP port (443)" from the UG has been removed.

10.2.1 HTTP Encoding

A client using the HTTPS profile SHALL:

- Use the POST request method
- Specify Content-Type: application/octet-stream
- Send one TTLV KMIP message in binary form in the body of each HTTP request
- Comply with the HTTP version claimed in the request line

A client using the HTTPS profile SHALL NOT:

- Use the HTTP Authorization header to transmit any authentication data

A server using the HTTPS profile SHALL:

- Return HTTP response code 200 Success if a KMIP response was returned, including KMIP error responses
- Specify Content-Type: application/octet-stream
- Send one TTLV KMIP message in binary form in the body of each HTTP response with response code 200
- Comply with the HTTP version claimed in the status line
- Send the Cache-Control: no-cache directive to prevent clients from caching KMIP responses (HTTP/1.1 only)

Servers supporting the OPTIONAL server-to-client Put and Notify messages SHALL behave as an HTTP client. Clients responding to these messages SHALL behave as an HTTP server.

A client MAY use the Accept-Encoding header to indicate it accepts compressed or otherwise transformed responses.

A clients MAY use HTTP/1.0 Keep Alive or HTTP/1.1 Connection headers to control persistent connection behavior.

The Content-Length header MAY be used by clients and servers to support persistent connections where allowed by HTTP. This header is not REQUIRED.

The Request-URI is not specified and MAY be used by clients and servers in an implementation specific fashion. The value / is RECOMMENDED.

KMIP servers supporting the HTTPS profile SHOULD listen on port TBD1. KMIP clients responding to OPTIONAL server-to-client Put and Notify messages should listen on port TBD2.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

11 Error Handling

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

1712 **11.1 General**

1713 These errors MAY occur when any protocol message is received by the server.

Error Definition	Action	Result Reason
Protocol major version mismatch	Response message containing a header and a Batch Item without Operation, but with the Result Status field set to Operation Failed	Invalid Message
Error parsing batch item or payload within batch item (e.g., REQUIRED fields missing, etc.)	Batch item fails; Result Status is Operation Failed	Invalid Message
The same field is contained in a header/batch item/payload more than once	Result Status is Operation Failed	Invalid Message
Same major version, different minor versions (e.g., client is newer); unknown fields/fields the server does not understand	Ignore unknown fields, process rest normally	N/A
Same major & minor version, unknown field	Result Status is Operation Failed	Invalid Field
Client is not allowed to perform the specified operation	Result Status is Operation Failed	Permission Denied
Operation 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

Deleted: required

1714 **Table 218: General Errors**

1715 **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 (e.g., initial date 5 years ago)	Operation Failed	Invalid Field
Error creating cryptographic object (e.g., key material generation issue)	Operation Failed	Cryptographic Failure
Trying to set more instances than the server supports of an attribute that	Operation Failed	Index Out of Bounds

MAY have multiple instances		
Trying to create a new object with the same Name attribute value as an existing object	Operation Failed	Invalid Field
The particular Application Namespace is not supported and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Template object is archived	Operation Failed	Object Archived

Table 219: Create Errors

1716

11.3 Create Key Pair

1717

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 (e.g., key material generation issue)	Operation Failed	Cryptographic Failure
Trying to create a new object with the same Name attribute value as an existing object	Operation Failed	Invalid Field
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

Deleted: Required

Table 220: Create Key Pair Errors

1718

1719 **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 (e.g., initial date 5 years ago)	Operation Failed	Invalid Field
Trying to register a new object with the same Name attribute value as an existing object	Operation Failed	Invalid Field
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

1720 **Table 221: Register Errors**

1721 **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 (e.g., not a symmetric key, or the permissions do not allow it)	Operation Failed	Permission Denied
Offset field is not permitted to be specified at the same time as any of the Activation Date, Process Start Date, Protect Stop Date, or Deactivation Date attributes	Operation Failed	Invalid Message
Cryptographic error during re-key	Operation Failed	Cryptographic Failure
The particular Application Namespace is not supported and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived

1722 **Table 222: Re-key Errors**

1723 **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 (e.g., initial date 5 years ago)	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

1724 **Table 223: Derive Key Errors**

1725 **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 (e.g., not a public key or the permissions do not allow it)	Operation Failed	Permission Denied
The Certificate Request does not contain a signed certificate request of the specified Certificate Request Type	Operation Failed	Invalid Field
Server does not support operation	Operation Failed	Operation Not Supported
The particular Application Namespace is not supported and Application Data cannot be generated if it was omitted	Operation Failed	Application Namespace Not Supported

from the client request		
Object is archived	Operation Failed	Object Archived

1726

Table 224: Certify Errors1727 **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 (e.g., not a certificate or the permissions do not allow it)	Operation Failed	Permission Denied
The Certificate Request does not contain a signed certificate request of the specified Certificate Request Type	Operation Failed	Invalid Field
Server does not support operation	Operation Failed	Operation Not Supported
Offset field 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

1728

Table 225: Re-certify Errors1729 **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 request	Operation Failed	Invalid Field

1730

Table 226: Locate Errors1731 **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

1732

Table 227: Check Errors

1733 **11.11 Get**

Error Definition	Result Status	Result Reason
Object does not exist	Operation Failed	Item Not Found
Wrapping key does not exist	Operation Failed	Item Not Found
Object with Wrapping Key ID exists, but it is not a key	Operation Failed	Illegal Operation
Object with Wrapping Key ID exists, but it is not able to be used for wrapping	Operation Failed	Permission Denied
Object with MAC/Signature Key ID exists, but it is not a key	Operation Failed	Illegal Operation
Object with MAC/Signature Key ID exists, but it is not able to be used for MACing/signing	Operation Failed	Permission Denied
Object exists and is not a Template, but the server only has attributes for this object	Operation Failed	Illegal Operation
Cryptographic Parameters associated with 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

1734 **Table 228: Get Errors**

1735 **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

1736 **Table 229: Get Attributes Errors**

1737 **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

1738 **Table 230: Get Attribute List Errors**

1739 **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 read-only attribute	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 is reached	Operation Failed	Index Out of Bounds
The particular Application Namespace is not supported and Application Data cannot be generated if it was omitted from the client request	Operation Failed	Application Namespace Not Supported
Object is archived	Operation Failed	Object Archived

Table 231: Add Attribute Errors

1740

1741 **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

Table 232: Modify Attribute Errors

1742

1743 **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 read-only/REQUIRED attribute	Operation Failed	Permission Denied
Attribute Index is specified, but attribute does not have multiple instances (i.e., no Attribute Index is permitted to be specified)	Operation Failed	Item Not Found
No attribute with specified name exists	Operation Failed	Item Not Found
Object is archived	Operation Failed	Object Archived

Deleted: required

1744 **Table 233: Delete Attribute Errors**

1745 **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

1746 **Table 234: Obtain Lease Errors**

1747 **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 object is not able to be used for protection purposes	Operation Failed	Illegal Operation
Both Usage Limits Byte Count and Usage Limits Object Count fields are specified	Operation Failed	Invalid Message
Neither Byte Count or Object Count is specified	Operation Failed	Invalid Message
A usage type (Byte Count or Object Count) is specified in the request, but the usage allocation for the object MAY only be given for the other type	Operation Failed	Operation Not Supported
Object is archived	Operation Failed	Object Archived

1748

Table 235: Get Usage Allocation Errors1749 **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 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

1750

Table 236: Activate Errors1751 **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 template or other object that is not able to be revoked	Operation Failed	Illegal Operation
Object is archived	Operation Failed	Object Archived

1752

Table 237: Revoke Errors1753 **11.21 Destroy**

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

1754

Table 238: Destroy Errors1755 **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

1756

Table 239: Archive Errors

1757 **11.23 Recover**

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

1758

Table 240: Recover Errors

1759 **11.24 Validate**

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

1760

Table 241: Validate Errors

1761 **11.25 Query**

1762 N/A

1763 **11.26 Cancel**

1764 N/A

1765 **11.27 Poll**

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

1766

Table 242: Poll Errors

1767 **11.28 Batch Items**

1768 These errors MAY occur when a protocol message with one or more batch items is processed by the
1769 server. If a message with one or more batch items was parsed correctly, then the response message
1770 SHOULD include response(s) to the batch item(s) in the request according to the table below.

1771

Error Definition	Result Status	Result Reason
Processing of batch item fails with Batch Error Continuation Option set to Stop	Batch item fails. Responses to batch items that have already been processed are returned normally. Responses to batch items that have not been processed are not returned.	See tables above, referring to the operation being performed in the batch item that failed

Processing of batch item fails with Batch Error Continuation Option set to Continue	Batch item fails. Responses to other batch items are returned normally.	See tables above, referring to the operation being performed in the batch item that failed
Processing of batch item fails with Batch Error Continuation Option set to Undo	Batch item fails. Batch items that had been processed have been undone and their responses are returned with Undone result status.	See tables above, referring to the operation being performed in the batch item that failed

Table 243: Batch Items Errors

1772

1773 **12 Security Considerations**

1774 TBD

1775

A. Attribute Cross-reference

1776 The following table of Attribute names indicates the Managed Object(s) for which each attribute applies.
1777 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 Length	x	x	x	x	x	x		
Cryptographic Parameters	x	x	x	x	x	x		
Certificate Type	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
Object Group	x	x	x	x	x	x	x	x
Link	x	x	x	x	x		x	

	Managed Object							
Application Specific Information	x	x	x	x	x	x	x	x
Contact Information	x	x	x	x	x	x	x	x
Last Changed Date	x	x	x	x	x	x	x	x
Custom Attribute	x	x	x	x	x	x	x	x

1778

Table 244: Attribute Cross-reference

1779

B. Tag Cross-reference

1780

This table is not normative.

Object	Defined	Type	Notes
Activation Date	3.17	Date-Time	
Application Data	3.28	Text String	
Application Namespace	3.28	Text String	
Application Specific Information	3.28	Structure	
Archive Date	3.25	Date-Time	
Asynchronous Correlation Value	6.8	Octet String	
Asynchronous Indicator	6.7	Boolean	
Attribute	2.1.1	Structure	
Attribute Index	2.1.1	Integer	
Attribute Name	2.1.1	Text String	
Attribute Value	2.1.1	*	type varies
Authentication	6.6	Structure	
Batch Count	6.14	Integer	
Batch Error Continuation Option	6.13 , 9.1.3.2.28	Enumeration	
Batch Item	6.15	Structure	
Batch Order Option	6.12	Boolean	
Block Cipher Mode	3.6 , 9.1.3.2.12	Enumeration	
Cancellation Result	4.25 , 9.1.3.2.23	Enumeration	
Certificate	2.2.1	Structure	
Certificate Issuer	3.8	Structure	
Certificate Request	4.6 , 4.7	Octet String	
Certificate Request Type	4.6 , 4.7 , 9.1.3.2.20	Enumeration	
Certificate Subject	3.9	Structure	
Certificate Subject Alternative Name	3.9	Text String	
Certificate Subject Distinguished Name	3.9	Text String	
Certificate Type	2.2.1 , 3.7 , 9.1.3.2.5	Enumeration	
Certificate Value	2.2.1	Octet String	
Common Template-Attribute	2.1.8	Structure	
Compromise Occurrence Date	0	Date-Time	
Compromise Date	3.23	Date-Time	
Contact Information	3.29	Text String	
Credential	2.1.2	Structure	
Credential Type	2.1.2 , 9.1.3.2.1	Enumeration	
Credential Value	2.1.2	Octet String	
Criticality Indicator	6.16	Boolean	

Object	Defined	Type	Notes
CRT Coefficient	2.1.7	Big Integer	
Cryptographic Algorithm	3.4 , 9.1.3.2.11	Enumeration	
Cryptographic Length	3.5	Integer	
Cryptographic Parameters	3.6	Structure	
Cryptographic Usage Mask	3.12 , 9.1.3.3.1	Integer	Bit mask
Custom Attribute	3.31	*	type varies
D	2.1.7	Big Integer	
Deactivation Date	3.20	Date-Time	
Derivation Data	4.5	Octet String	
Derivation Method	4.5 , 9.1.3.2.19	Enumeration	
Derivation Parameters	4.5	Structure	
Destroy Date	3.21	Date-Time	
Digest	3.10	Structure	
Digest Value	3.10	Octet String	
Encryption Key Information	2.1.5	Structure	
Extensions	9.1.3		
G	2.1.7	Big Integer	
Hashing Algorithm	3.6 , 3.10 , 9.1.3.2.14	Enumeration	
Initial Date	3.16	Date-Time	
Initialization Vector	4.5	Octet String	
Issuer	3.8	Text String	
Iteration Count	4.5	Integer	
IV/Counter/Nonce	2.1.5	Octet String	
J	2.1.7	Big Integer	
Key	2.1.7	Octet String	
Key Block	2.1.3	Structure	
Key Material	2.1.4 , 2.1.7	Octet String / Structure	
Key Part Identifier	2.2.5	Integer	
Key Value	2.1.4	Octet String / Structure	
Key Value Type	2.1.4 , 9.1.3.2.2	Enumeration	
Key Wrapping Data	2.1.5	Structure	
Key Wrapping Specification	2.1.6	Structure	
Last Changed Date	3.30	Date-Time	
Lease Time	3.13	Interval	
Link	3.27	Structure	
Link Type	3.27 , 9.1.3.2.18	Enumeration	
Linked Object Identifier	3.27	Text String	

Object	Defined	Type	Notes
MAC/Signature	2.1.5	Octet String	
MAC/Signature Key Information	2.1.5	Text String	
Maximum Items	4.8	Integer	
Maximum Response Size	6.3	Integer	
Message Extension	6.16	Structure	
Modulus	2.1.7	Big Integer	
Name	3.2	Structure	
Name Type	3.2 , 9.1.3.2.9	Enumeration	
Name Value	3.2	Text String	
Object Group	3.26	Text String	
Object Type	3.3 , 9.1.3.2.10	Enumeration	
Offset	4.4 , 4.7	Interval	
Opaque Data Type	2.2.8 , 9.1.3.2.8	Enumeration	
Opaque Data Value	2.2.8	Octet String	
Opaque Object	2.2.8	Structure	
Operation	6.2 , 9.1.3.2.25	Enumeration	
Operation Policy Name	3.11	Text String	
P	2.1.7	Big Integer	
Padding Method	3.6 , 9.1.3.2.13	Enumeration	
Prime Exponent P	2.1.7	Big Integer	
Prime Exponent Q	2.1.7	Big Integer	
Prime Field Size	2.2.5	Big Integer	
Private Exponent	2.1.7	Big Integer	
Private Key	2.2.4	Structure	
Private Key Template-Attribute	2.1.8	Structure	
Private Key Unique Identifier	4.2	Text String	
Process Start Date	3.18	Date-Time	
Protect Stop Date	3.19	Date-Time	
Protocol Version	6.1	Structure	
Protocol Version Major	6.1	Integer	
Protocol Version Minor	6.1	Integer	
Public Exponent	2.1.7	Big Integer	
Public Key	2.2.3	Structure	
Public Key Template-Attribute	2.1.8	Structure	
Public Key Unique Identifier	4.2	Text String	
Put Function	5.2 , 9.1.3.2.24	Enumeration	
Q	2.1.7	Big Integer	
Q String	2.1.7	Octet String	

Object	Defined	Type	Notes
Query Function	4.24 , 9.1.3.2.22	Enumeration	
Recommended Curve	2.1.7 , 9.1.3.2.4	Enumeration	
Replaced Unique Identifier	5.2	Text String	
Request Header	7.2 , 7.3	Structure	
Request Message	7.1	Structure	
Request Payload	4 , 5 , 7.2 , 7.3	Structure	
Response Header	7.2 , 7.3	Structure	
Response Message	7.1	Structure	
Response Payload	4 , 7.2 , 7.3	Structure	
Result Message	6.11	Text String	
Result Reason	6.10 , 9.1.3.2.27	Enumeration	
Result Status	6.9 , 9.1.3.2.26	Enumeration	
Revocation Message	3.24	Text String	
Revocation Reason	3.24	Structure	
Revocation Reason Code	3.24 , 9.1.3.2.17	Enumeration	
Role Type	3.6 , 9.1.3.2.15	Enumeration	
Salt	4.5	Octet String	
Secret Data	2.2.7	Structure	
Secret Data Type	2.2.7 , 9.1.3.2.7	Enumeration	
Serial Number	3.8	Text String	
Server Information	4.24	Structure	contents vendor-specific
Split Key	2.2.5	Structure	
Split Key Method	2.2.5 , 9.1.3.2.6	Enumeration	
Split Key Parts	2.2.5	Integer	
Split Key Threshold	2.2.5	Integer	
State	3.15 , 9.1.3.2.16	Enumeration	
Storage Status Mask	4.8 , 9.1.3.3.2	Integer	Bit mask
Symmetric Key	2.2.2	Structure	
Template	2.2.6	Structure	
Template-Attribute	2.1.8	Structure	
Time Stamp	6.5	Date-Time	
Transparent*	2.1.7	Structure	
Unique Identifier	3.1	Text String	
Unique Batch Item ID	6.4	Octet String	
Usage Limits	3.14	Structure	
Usage Limits Byte Count	3.14	Big Integer	
Usage Limits Object Count	3.14	Big Integer	
Usage Limits Total Bytes	3.14	Big Integer	

Object	Defined	Type	Notes
Usage Limits Total Objects	3.14	Big Integer	
Validity Date	4.23	Date-Time	
Validity Indicator	4.23 , 9.1.3.2.21	Enumeration	
Vendor Extension	6.16	Structure	contents vendor-specific
Vendor Identification	4.24 , 6.16	Text String	
Wrapping Method	2.1.5 , 9.1.3.2.3	Enumeration	
X	2.1.7	Big Integer	
Y	2.1.7	Big Integer	

Table 245: Tag Cross-reference

1781

1782
1783
1784

C. Operation and Object Cross-reference

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

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

1785

Table 246: Operation and Object Cross-reference

1786 **D. Acronyms**

1787 The following abbreviations and acronyms are used in this document:

- 1788 3DES - Three key Data Encryption Standard
- 1789 AES - Advanced Encryption Standard specified in FIPS 197
- 1790 ASN.1 - Abstract Syntax Notation One
- 1791 CA - Certification Authority
- 1792 CBC - Cipher Block Chaining
- 1793 CPU - Central Processing Unit
- 1794 CRL - Certificate Revocation List
- 1795 CRT - Chinese Remainder Theorem
- 1796 DER - Distinguished Encoding Rules
- 1797 DES - Data Encryption Standard
- 1798 DH - Diffie-Hellman
- 1799 DSA - Digital Signature Algorithm specified in FIPS 186-3
- 1800 DSKPP - Dynamic Symmetric Key Provisioning Protocol
- 1801 ECB - Electronic Code Book
- 1802 ECDH - Elliptic Curve Diffie-Hellman
- 1803 ECDSA - Elliptic Curve Digital Signature Algorithm specified in ANSX9.62
- 1804 HMAC - Keyed-Hash Message Authentication Code specified in FIPS 198
- 1805 HTTP - Hyper Text Transfer Protocol
- 1806 HTTP(S) - Hyper Text Transfer Protocol (Secure socket)
- 1807 IEEE - Institute of Electrical and Electronics Engineers
- 1808 IETF - Internet Engineering Task Force
- 1809 IPsec - Internet Protocol Security
- 1810 IV - Initialization Vector
- 1811 KMIP - Key Management Interoperability Protocol
- 1812 MAC - Message Authentication Code
- 1813 MD5 - Message Digest 5 Algorithm
- 1814 PBKDF2 - Password-Based Key Derivation Function 2
- 1815 PGP - Pretty Good Privacy
- 1816 PKCS - Public Key Cryptography Standards
- 1817 POSIX - Portable Operating System Interface
- 1818 RFC - Request for Comments documents of IETF
- 1819 RSA - Rivest, Shamir, Adelman (an algorithm)
- 1820 SHA-1 - Secure Hash Algorithm Revision One
- 1821 SSL/TLS - Secure Sockets Layer/Transport Layer Security
- 1822 S/MIME - Secure/Multipurpose Internet Mail Extensions

- 1823 TCP - Transport Control Protocol
- 1824 TTLV - Tag, Type, Length, Value
- 1825 URI - Unique Resource Identifier
- 1826 UTF - Universal Transformation Format
- 1827 XML - Extensible Markup Language

E. List of Figures and Tables

1829 Figures

1830	Figure 1: Cryptographic Object States and Transitions	36
------	---	----

1831

1832 Tables

1833	Table 1: Attribute Object Structure	10
1834	Table 2: Credential Object Structure	11
1835	Table 3: Key Block Object Structure	12
1836	Table 4: Key Value Object Structure	12
1837	Table 5: Key Wrapping Data Object Structure	13
1838	Table 6: Encryption Key Information Object Structure	14
1839	Table 7: MAC/Signature Key Information Object Structure	14
1840	Table 8: Key Wrapping Specification Object Structure	14
1841	Table 9: Key Material Object Structure for Transparent Symmetric Keys	15
1842	Table 10: Key Material Object Structure for Transparent DSA Private Keys	15
1843	Table 11: Key Material Object Structure for Transparent DSA Public Keys	15
1844	Table 12: Key Material Object Structure for Transparent RSA Private Keys	16
1845	Table 13: Key Material Object Structure for Transparent RSA Public Keys	16
1846	Table 14: Key Material Object Structure for Transparent DH Private Keys	16
1847	Table 15: Key Material Object Structure for Transparent DH Public Keys	17
1848	Table 16: Key Material Object Structure for Transparent ECDSA Private Keys	17
1849	Table 17: Key Material Object Structure for Transparent ECDSA Public Keys	17
1850	Table 18: Key Material Object Structure for Transparent ECDH Private Keys	18
1851	Table 19: Key Material Object Structure for Transparent ECDH Public Keys	18
1852	Table 20: Template-Attribute Object Structure	18
1853	Table 21: Certificate Object Structure	19
1854	Table 22: Symmetric Key Object Structure	19
1855	Table 23: Public Key Object Structure	19
1856	Table 24: Private Key Object Structure	19
1857	Table 25: Split Key Object Structure	20
1858	Table 26: Template Object Structure	21
1859	Table 27: Secret Data Object Structure	22
1860	Table 28: Opaque Object Structure	22
1861	Table 29: Unique Identifier Attribute	23
1862	Table 30: Unique Identifier Attribute Rules	23
1863	Table 31: Name Attribute Structure	24
1864	Table 32: Name Attribute Rules	24
1865	Table 33: Object Type Attribute	24
1866	Table 34: Object Type Attribute Rules	24
1867	Table 35: Cryptographic Algorithm Attribute	25

1868	Table 36: Cryptographic Algorithm Attribute Rules	25
1869	Table 37: Cryptographic Length Attribute	25
1870	Table 38: Cryptographic Length Attribute Rules	25
1871	Table 39: Cryptographic Parameters Attribute Structure	26
1872	Table 40: Cryptographic Parameters Attribute Rules.....	26
1873	Table 41: Role Types	27
1874	Table 42: Certificate Type Attribute	27
1875	Table 43: Certificate Type Attribute Rules	28
1876	Table 44: Certificate Issuer Attribute Structure	28
1877	Table 45: Certificate Issuer Attribute Rules	28
1878	Table 46: Certificate Subject Attribute Structure	29
1879	Table 47: Certificate Subject Attribute Rules	29
1880	Table 48: Digest Attribute Structure.....	29
1881	Table 49: Digest Attribute Rules	30
1882	Table 50: Operation Policy Name Attribute.....	30
1883	Table 51: Operation Policy Name Attribute Rules.....	30
1884	Table 52: Default Operation Policy for Secret Objects.....	31
1885	Table 53: Default Operation Policy for Certificates and Public Key Objects	32
1886	Table 54: Default Operation Policy for Private Template Objects	32
1887	Table 55: Default Operation Policy for Public Template Objects	33
1888	Table 56: X.509 Key Usage to Cryptographic Usage Mask Mapping	34
1889	Table 57: Cryptographic Usage Mask Attribute	34
1890	Table 58: Cryptographic Usage Mask Attribute Rules	34
1891	Table 59: Lease Time Attribute.....	35
1892	Table 60: Lease Time Attribute Rules	35
1893	Table 61: Usage Limits Attribute Structure	36
1894	Table 62: Usage Limits Attribute Rules	36
1895	Table 63: State Attribute.....	37
1896	Table 64: State Attribute Rules.....	38
1897	Table 65: Initial Date Attribute	38
1898	Table 66: Initial Date Attribute Rules	38
1899	Table 67: Activation Date Attribute	38
1900	Table 68: Activation Date Attribute Rules	39
1901	Table 69: Process Start Date Attribute	39
1902	Table 70: Process Start Date Attribute Rules	39
1903	Table 71: Protect Stop Date Attribute	40
1904	Table 72: Protect Stop Date Attribute Rules	40
1905	Table 73: Deactivation Date Attribute	40
1906	Table 74: Deactivation Date Attribute Rules	40
1907	Table 75: Destroy Date Attribute	41
1908	Table 76: Destroy Date Attribute Rules	41
1909	Table 77: Compromise Occurrence Date Attribute	41

1910	Table 78: Compromise Occurrence Date Attribute Rules	41
1911	Table 79: Compromise Date Attribute	42
1912	Table 80: Compromise Date Attribute Rules	42
1913	Table 81: Revocation Reason Attribute Structure	42
1914	Table 82: Revocation Reason Attribute Rules	42
1915	Table 83: Archive Date Attribute	43
1916	Table 84: Archive Date Attribute Rules	43
1917	Table 85: Object Group Attribute	43
1918	Table 86: Object Group Attribute Rules	43
1919	Table 87: Link Attribute Structure	44
1920	Table 88: Link Attribute Structure Rules	45
1921	Table 89: Application Specific Information Attribute	45
1922	Table 90: Application Specific Information Attribute Rules	45
1923	Table 91: Contact Information Attribute	45
1924	Table 92: Contact Information Attribute Rules	46
1925	Table 93: Last Changed Date Attribute	46
1926	Table 94: Last Changed Date Attribute Rules	46
1927	Table 95 Custom Attribute	47
1928	Table 96: Custom Attribute Rules	47
1929	Table 97: Create Request Payload	48
1930	Table 98: Create Response Payload	48
1931	Table 99: Create Attribute Requirements	48
1932	Table 100: Create Key Pair Request Payload	49
1933	Table 101: Create Key Pair Response Payload	50
1934	Table 102: Create Key Pair Attribute Requirements	50
1935	Table 103: Register Request Payload	50
1936	Table 104: Register Response Payload	51
1937	Table 105: Register Attribute Requirements	51
1938	Table 106: Computing New Dates from Offset during Re-key	52
1939	Table 107: Re-key Attribute Requirements	52
1940	Table 108: Re-key Request Payload	53
1941	Table 109: Re-key Response Payload	53
1942	Table 110: Derive Key Request Payload	54
1943	Table 111: Derive Key Response Payload	55
1944	Table 112: Derivation Parameters Structure (Except PBKDF2)	55
1945	Table 113: PBKDF2 Derivation Parameters Structure	56
1946	Table 114: Certify Request Payload	56
1947	Table 115: Certify Response Payload	57
1948	Table 116: Computing New Dates from Offset during Re-certify	57
1949	Table 117: Re-certify Attribute Requirements	58
1950	Table 118: Re-certify Request Payload	58
1951	Table 119: Re-certify Response Payload	59

1952	Table 120: Locate Request Payload.....	60
1953	Table 121: Locate Response Payload.....	60
1954	Table 122: Check Request Payload.....	61
1955	Table 123: Check Response Payload.....	62
1956	Table 124: Get Request Payload.....	62
1957	Table 125: Get Response Payload.....	62
1958	Table 126: Get Attributes Request Payload.....	63
1959	Table 127: Get Attributes Response Payload.....	63
1960	Table 128: Get Attribute List Request Payload.....	63
1961	Table 129: Get Attribute List Response Payload.....	63
1962	Table 130: Add Attribute Request Payload.....	64
1963	Table 131: Add Attribute Response Payload.....	64
1964	Table 132: Modify Attribute Request Payload.....	64
1965	Table 133: Modify Attribute Response Payload.....	64
1966	Table 134: Delete Attribute Request Payload.....	65
1967	Table 135: Delete Attribute Response Payload.....	65
1968	Table 136: Obtain Lease Request Payload.....	65
1969	Table 137: Obtain Lease Response Payload.....	66
1970	Table 138: Get Usage Allocation Request Payload.....	66
1971	Table 139: Get Usage Allocation Response Payload.....	67
1972	Table 140: Activate Request Payload.....	67
1973	Table 141: Activate Response Payload.....	67
1974	Table 142: Revoke Request Payload.....	67
1975	Table 143: Revoke Response Payload.....	67
1976	Table 144: Destroy Request Payload.....	68
1977	Table 145: Destroy Response Payload.....	68
1978	Table 146: Archive Request Payload.....	68
1979	Table 147: Archive Response Payload.....	68
1980	Table 148: Recover Request Payload.....	69
1981	Table 149: Recover Response Payload.....	69
1982	Table 150: Validate Request Payload.....	69
1983	Table 151: Validate Response Payload.....	69
1984	Table 152: Query Request Payload.....	70
1985	Table 153: Query Response Payload.....	71
1986	Table 154: Cancel Request Payload.....	71
1987	Table 155: Cancel Response Payload.....	71
1988	Table 156: Poll Request Payload.....	72
1989	Table 157: Notify Message Payload.....	72
1990	Table 158: Put Message Payload.....	73
1991	Table 159: Protocol Version Structure in Message Header.....	74
1992	Table 160: Operation in Batch Item.....	74
1993	Table 161: Maximum Response Size in Message Request Header.....	74

1994	Table 162: Unique Batch Item ID in Batch Item.....	74
1995	Table 163: Time Stamp in Message Header	74
1996	Table 164: Authentication Structure in Message Header.....	75
1997	Table 165: Asynchronous Indicator in Message Request Header	75
1998	Table 166: Asynchronous Correlation Value in Response Batch Item.....	75
1999	Table 167: Result Status in Response Batch Item.....	76
2000	Table 168: Result Reason in Response Batch Item	76
2001	Table 169: Result Message in Response Batch Item	76
2002	Table 170: Batch Order Option in Message Request Header.....	77
2003	Table 171: Batch Error Continuation Option in Message Request Header	77
2004	Table 172: Batch Count in Message Header.....	77
2005	Table 173: Batch Item in Message	78
2006	Table 174: Message Extension Structure in Batch Item	78
2007	Table 175: Request Message Structure	78
2008	Table 176: Response Message Structure.....	78
2009	Table 177: Synchronous Request Header Structure	79
2010	Table 178: Synchronous Request Batch Item Structure	79
2011	Table 179: Synchronous Response Header Structure.....	79
2012	Table 180: Synchronous Response Batch Item Structure	80
2013	Table 181: Asynchronous Request Header Structure.....	80
2014	Table 182: Asynchronous Request Batch Item Structure	81
2015	Table 183: Asynchronous Response Header Structure.....	81
2016	Table 184: Asynchronous Response Batch Item Structure	81
2017	Table 185: Allowed Item Type Values	83
2018	Table 186: Allowed Item Length Values	83
2019	Table 187: Tag Values	90
2020	Table 188: Credential Type Enumeration	91
2021	Table 189: Key Value Type Enumeration	91
2022	Table 190: Wrapping Method Enumeration	92
2023	Table 191: Recommended Curve Enumeration for ECDSA and ECDH	92
2024	Table 192: Certificate Type Enumeration	93
2025	Table 193: Split Key Method Enumeration	93
2026	Table 194: Secret Data Type Enumeration.....	93
2027	Table 195: Opaque Data Type Enumeration	93
2028	Table 196: Name Type Enumeration.....	93
2029	Table 197: Object Type Enumeration	94
2030	Table 198: Cryptographic Algorithm Enumeration	94
2031	Table 199: Block Cipher Mode Enumeration	95
2032	Table 200: Padding Method Enumeration	95
2033	Table 201: Hashing Algorithm Enumeration	96
2034	Table 202: Role Type Enumeration	97
2035	Table 203: State Enumeration.....	98

2036	Table 204: Revocation Reason Code Enumeration.....	98
2037	Table 205: Link Type Enumeration.....	98
2038	Table 206: Derivation Method Enumeration.....	99
2039	Table 207: Certificate Request Type Enumeration.....	99
2040	Table 208: Validity Indicator Enumeration.....	99
2041	Table 209: Query Function Enumeration.....	100
2042	Table 210: Cancellation Result Enumeration.....	100
2043	Table 211: Put Function Enumeration.....	100
2044	Table 212: Operation Enumeration.....	101
2045	Table 213: Result Status Enumeration.....	102
2046	Table 214: Result Reason Enumeration.....	102
2047	Table 215: Batch Error Continuation Enumeration.....	103
2048	Table 216: Cryptographic Usage Mask.....	103
2049	Table 217: Storage Status Mask.....	104
2050	Table 218: General Errors.....	106
2051	Table 219: Create Errors.....	107
2052	Table 220: Create Key Pair Errors.....	107
2053	Table 221: Register Errors.....	108
2054	Table 222: Re-key Errors.....	108
2055	Table 223: Derive Key Errors.....	109
2056	Table 224: Certify Errors.....	110
2057	Table 225: Re-certify Errors.....	110
2058	Table 226: Locate Errors.....	110
2059	Table 227: Check Errors.....	110
2060	Table 228: Get Errors.....	111
2061	Table 229: Get Attributes Errors.....	111
2062	Table 230: Get Attribute List Errors.....	111
2063	Table 231: Add Attribute Errors.....	112
2064	Table 232: Modify Attribute Errors.....	112
2065	Table 233: Delete Attribute Errors.....	113
2066	Table 234: Obtain Lease Errors.....	113
2067	Table 235: Get Usage Allocation Errors.....	114
2068	Table 236: Activate Errors.....	114
2069	Table 237: Revoke Errors.....	114
2070	Table 238: Destroy Errors.....	114
2071	Table 239: Archive Errors.....	115
2072	Table 240: Recover Errors.....	115
2073	Table 241: Validate Errors.....	115
2074	Table 242: Poll Errors.....	115
2075	Table 243: Batch Items Errors.....	116
2076	Table 244: Attribute Cross-reference.....	118
2077	Table 245: Tag Cross-reference.....	123

2078 Table 246: Operation and Object Cross-reference 124
2079

2080 F. Acknowledgements

2081 The following individuals have participated in the creation of this specification and are gratefully
2082 acknowledged:

2083 **Original Authors of the initial contribution:**

2084 David Babcock, HP
2085 Steven Bade, IBM
2086 Paolo Bezoari, NetApp
2087 Mathias Björkqvist, IBM
2088 Bruce Brinson, EMC
2089 Christian Cachin, IBM
2090 Tony Crossman, Thales/nCipher
2091 Stan Feather, HP
2092 Indra Fitzgerald, HP
2093 Judy Furlong, EMC
2094 Jon Geater, Thales/nCipher
2095 Bob Griffin, EMC
2096 Robert Haas, IBM (editor)
2097 Timothy Hahn, IBM
2098 Jack Harwood, EMC
2099 Walt Hubis, LSI
2100 Glen Jaquette, IBM
2101 Jeff Kravitz, IBM (editor emeritus)
2102 Michael McIntosh, IBM
2103 Brian Metzger, HP
2104 Anthony Nadalin, IBM
2105 Elaine Palmer, IBM
2106 Joe Pato, HP
2107 René Pawlitzek, IBM
2108 Subhash Sankuratripati, NetApp
2109 Mark Schiller, HP
2110 Martin Skagen, Brocade
2111 Marcus Streets, Thales/nCipher
2112 John Tattan, EMC
2113 Karla Thomas, Brocade
2114 Marko Vukolić, IBM
2115 Steve Wierenga, HP

2116 **Participants:**

2117 TBD

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.