



# TBD (Key Management Interoperability Protocol)

**Editor's Draft 0.98**

**21 May 2009**

**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  
Anthony Nadalin

**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 .....	8
1.7 Non-normative References .....	9
1.8 Compliance .....	9
2 Objects .....	9
2.1 Base Objects .....	9
2.1.1 Attribute .....	9
2.1.2 Credential .....	10
2.1.3 Key Block .....	10
2.1.4 Key Value .....	11
2.1.5 Key Wrapping Data .....	12
2.1.6 Key Wrapping Specification .....	13
2.1.7 Transparent Key Structures .....	13
2.1.8 Template-Attribute Structures .....	16
2.2 Managed Objects .....	17
2.2.1 Certificate .....	17
2.2.2 Symmetric Key .....	17
2.2.3 Public Key .....	17
2.2.4 Private Key .....	17
2.2.5 Split Key .....	18
2.2.6 Template .....	19
2.2.7 Policy Template .....	19
2.2.8 Secret Data .....	20
2.2.9 Opaque Object .....	20
3 Attributes .....	21
3.1 Unique Identifier .....	21
3.2 Name .....	21
3.3 Object Type .....	22
3.4 Cryptographic Algorithm .....	22
3.5 Cryptographic Length .....	23
3.6 Cryptographic Parameters .....	23
3.7 Certificate Type .....	24
3.8 Certificate Issuer .....	25
3.9 Certificate Subject .....	25
3.10 Digest .....	26
3.11 Operation Policy Name .....	26
3.11.1 Operations outside of operation policy control .....	27
3.11.2 Default Operation Policy .....	27

3.12 Cryptographic Usage Mask .....	30
3.13 Lease Time .....	31
3.14 Usage Limits .....	31
3.15 State.....	33
3.16 Initial Date .....	34
3.17 Activation Date .....	35
3.18 Process Start Date .....	35
3.19 Protect Stop Date .....	36
3.20 Deactivation Date .....	36
3.21 Destroy Date .....	37
3.22 Compromise Occurrence Date .....	37
3.23 Compromise Date .....	38
3.24 Revocation Reason .....	38
3.25 Archive Date.....	39
3.26 Object Group.....	39
3.27 Link .....	40
3.28 Application Specific Identification .....	41
3.29 Contact Information .....	42
3.30 Last Changed Date .....	42
3.31 Custom Attribute .....	43
4 Client-to-Server Operations .....	43
4.1 Create .....	44
4.2 Create Key Pair.....	45
4.3 Register.....	47
4.4 Re-key.....	49
4.5 Derive Key .....	51
4.6 Certify.....	53
4.7 Re-certify.....	54
4.8 Locate .....	56
4.9 Check.....	58
4.10 Get .....	60
4.11 Get Attributes .....	60
4.12 Get Attribute List .....	61
4.13 Add Attribute .....	61
4.14 Modify Attribute .....	62
4.15 Delete Attribute .....	63
4.16 Obtain Lease.....	63
4.17 Get Usage Allocation.....	64
4.18 Activate .....	65
4.19 Revoke.....	66
4.20 Destroy.....	66
4.21 Archive .....	67
4.22 Recover.....	67
4.23 Validate .....	68
4.24 Query .....	68

4.25 Cancel.....	70
4.26 Poll.....	70
5 Server-to-Client Operations.....	71
5.1 Notify.....	71
5.2 Put.....	71
6 Message Contents.....	72
6.1 Protocol Version.....	72
6.2 Operation.....	73
6.3 Maximum Response Size.....	73
6.4 Unique Message ID.....	73
6.5 Time Stamp.....	73
6.6 Authentication.....	73
6.7 Asynchronous Indicator.....	74
6.8 Asynchronous Correlation Value.....	74
6.9 Result Status.....	74
6.10 Result Reason.....	74
6.11 Result Message.....	75
6.12 Batch Order Option.....	75
6.13 Batch Error Continuation Option.....	75
6.14 Batch Count.....	76
6.15 Batch Item.....	76
6.16 Message Extension.....	76
7 Message Format.....	76
7.1 Message Structure.....	77
7.2 Synchronous Operations.....	77
7.3 Asynchronous Operations.....	78
8 Authentication.....	79
9 Message Encoding.....	80
9.1 TTLV Encoding.....	80
9.1.1 TTLV Encoding Fields.....	80
9.1.2 Examples.....	82
9.1.3 Defined Values.....	83
9.2 XML Encoding.....	100
10 Transport.....	100
11 Error Handling.....	101
11.1 General.....	101
11.2 Create.....	102
11.3 Create Key Pair.....	102
11.4 Register.....	102
11.5 Re-key.....	103
11.6 Derive Key.....	103
11.7 Certify.....	104
11.8 Re-certify.....	104
11.9 Locate.....	104
11.10 Check.....	104

11.11 Get .....	105
11.12 Get Attributes .....	105
11.13 Get Attribute List .....	105
11.14 Add Attribute .....	106
11.15 Modify Attribute .....	106
11.16 Delete Attribute .....	106
11.17 Obtain Lease.....	107
11.18 Get Usage Allocation.....	107
11.19 Activate .....	107
11.20 Revoke.....	108
11.21 Destroy.....	108
11.22 Archive .....	108
11.23 Recover.....	108
11.24 Validate .....	108
11.25 Query .....	108
11.26 Cancel.....	109
11.27 Poll.....	109
11.28 Batch Items .....	109
12 Security Considerations.....	109
A. Attribute Cross-reference .....	110
B. Tag Cross-reference .....	112
C. Operation and Object Cross-reference .....	117
D. Acronyms.....	118
E. Acknowledgements .....	120
F. Revision History .....	121

---

# 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 will be 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 which can be obtained from the system via operations. Certain attributes may be changed, added or deleted, again 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 in proxying certificate management functions through a key server.

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:

- 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

## 1.6 Normative References

TBD



38 **1.7 Non-normative References**

39 TBD

40 **1.8 Compliance**

41 TBD

42 **2 Objects**

43 The following subsections describe the objects that are passed between the clients and servers of the key  
44 management system. Some of these object types, called *Base Objects*, are used only in the protocol  
45 itself, and are not considered Managed Objects. Key management systems may choose to support a  
46 subset of the Managed Objects. The object descriptions refer to the primitive data types they are  
47 composed of. These primitive data types are

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

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

58 **2.1 Base Objects**

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

61 **2.1.1 Attribute**

62 An object, used for sending and receiving Managed Object attributes. The *Attribute Name*  
63 is a text-string which is used to identify the attribute. The *Attribute Index* is an index  
64 number assigned by the key management server when a specified named attribute is  
65 allowed to have multiple instances. The index number is used to identify the particular  
66 instance. Index numbers start with 0. The index number of an attribute is never changed  
67 when other instances are added or deleted. For example, if a particular attribute has 4  
68 instances with index numbers 0, 1, 2 and 3, and the instance with index 2 is deleted, the  
69 index number of instance 3 is not changed. Attributes which have a single instance have  
70 an Attribute Index of 0, which is assumed if the index is not specified. The *Attribute Value*  
71 is either a primitive data type, or structured object, depending on the attribute.

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

72  
73  
74  
75

### 2.1.2 Credential

A credential is a protocol-only object, used for client identification purposes and not managed by the key management system, e.g., user id/password pairs, Kerberos tokens, etc. See Section 8 .

Object	Encoding	Required
Credential	Structure	Yes
Credential Type	Enumeration	Yes
Credential Value	Octet String	Yes

76

### 2.1.3 Key Block

77  
78  
79  
80

A *Key Block* object encapsulates all of the information that is closely associated with a cryptographic key. A Key Block object may contain different information depending on who it is sent to and when it is sent. It contains a Key Value of one of the following *Key Value Types*:

81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93

- *Raw* – This is a key which consists of “pure” cryptographic key material, encoded as a string of bytes.
- *Opaque* – This is an encoded key for which the encoding is unknown to the key management system. It is encoded as a string of bytes.
- *PKCS1* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#1 object.
- *PKCS8* – This is an encoded private key, expressed as a DER-encoded ASN.1 PKCS#8 object, supporting both RSAPrivateKey syntax and EncryptedPrivateKey.
- *Several Transparent Key types* – These are algorithm-specific structures containing defined values for the various key types, as defined in Section 2.1.6 .
- *Extensions* – These are vendor-specific extensions to allow for proprietary or legacy key formats.

94  
95

It contains also the Cryptographic Algorithm and the Cryptographic Length. Some example values are:

96  
97  
98

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

99  
100

The Key Block may optionally contain a Key Wrapping Data structure, which indicates that the key is wrapped, or MAC'ed/signed, or both.

Object	Encoding	Required Field
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 encapsulated in the Key Value. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Length below must also be present.
Cryptographic Length	Integer	Yes, may be omitted only if this information is encapsulated in the Key Value. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Algorithm above must also be present.
Key Wrapping Data	Structure	No

101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111

### 2.1.4 Key Value

The *Key Value* is used only inside a Key Block and is either an Octet String or a structure:

- 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 with 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 may be wrapped, signed or MAC'ed along with the key material itself.
- The Key Value Octet String is the wrapped TTLV-encoded (see Section 9.1 ) Key Value structure.

Object	Encoding	Required Field
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

112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135

## 2.1.5 Key Wrapping Data

The Key Block may also supply optional information about a cryptographic key wrapping mechanism used to wrap the Key Value. This consists of a *Key Wrapping Data* structure.

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 for the encryption key.
- A *MAC/Signature Key Information* with the Unique Identifier value for the MAC'ing or signing key.
- A *MAC/Signature* field with the MAC or signature of the Key Value.
- An *IV/Counter/Nonce* if required by the wrapping method.

If wrapping is used, the whole Key Value structure is wrapped with the wrapping key material unless otherwise specified by the Wrapping Method. The algorithm is determined by the Cryptographic Algorithm attribute set for the key. Similarly, the Cryptographic Parameters attribute of the key will identify the mode of operation or hashing algorithm to be used.

The following wrapping methods are currently defined:

- *Encrypt* only (possibly includes authenticated encryption algorithms that use a single key)
- *MAC/sign* only
- *Encrypt then MAC/sign*
- *MAC/sign then encrypt*
- *TR-31*
- *Extensions*

Object	Encoding	Required Field
Key Wrapping Data	Structure	Yes
Wrapping Method	Enumeration	Yes
Encryption Key Information	Structure	No
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

136  
137

The structures of the Encryption Key Information and the MAC/Signature Key Information are as follows:

Object	Encoding	Required Field
Encryption Key Information	Structure	Yes
Unique Identifier	Text string	Yes
Cryptographic	Structure	No

138

Parameters		
------------	--	--

Object	Encoding	Required Field
MAC/Signature Key Information	Structure	Yes
Unique Identifier	Text string	Yes. It can be the Unique Identifier of the Private or of the Public Key
Cryptographic Parameters	Structure	No

139

### 2.1.6 Key Wrapping Specification

140

This is a separate structure defined for operations that provide the option to return wrapped keys. The *Key Wrapping Specification* must be specified inside the operation request, if clients wish 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 can verify that they match one of the instances of the Cryptographic Parameters attribute of the corresponding key. If Cryptographic Parameters are omitted, the server can choose to use the Cryptographic Parameters attribute with the lowest index of the corresponding key. If the corresponding key does not have any Cryptographic Parameters attribute, or if no match is found, an error is returned.

141

142

143

144

145

146

147

148

This structure contains :

149

150

- A Wrapping Method that indicates the method used to wrap the Key Value.

151

152

- An Encryption Key Information with the Unique Identifier value of the encryption key and associated cryptographic parameters.

153

154

- A MAC/Signature Key Information with the Unique Identifier value of the MAC'ing or signing key and associated cryptographic parameters.

155

156

- Zero or more Attribute Names to indicate the attributes to be wrapped with the key material.

Object	Encoding	Required Field
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

157

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

158

159

### 2.1.7 Transparent Key Structures

160

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

161

162  
163  
164

### 2.1.7.1 Transparent Symmetric Key

If the Key Value Type in the Key Block is *Transparent Symmetric Key*, then Key Material is a structure as follows:

Object	Encoding	Required Field
Key Material	Structure	Yes
Key	Octet String	Yes

165  
166  
167

### 2.1.7.2 Transparent DSA Private Key

If the Key Value Type in the Key Block is *Transparent DSA Private Key*, then Key Material is a structure as follows:

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

168  
169  
170

### 2.1.7.3 Transparent DSA Public Key

If the Key Value Type in the Key Block is *Transparent DSA Public Key*, then Key Material is a structure as follows:

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

171  
172  
173

### 2.1.7.4 Transparent RSA Private Key

If the Key Value Type in the Key Block is *Transparent RSA Private Key*, then Key Material is a structure as follows:

Object	Encoding	Required Field
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

174 Note: One of the following must be present:

- 175 • Private Exponent
- 176 • P and Q
- 177 • Prime Exponent P and Prime Exponent Q.

### 178 **2.1.7.5 Transparent RSA Public Key**

179 If the Key Value Type in the Key Block is *Transparent RSA Public Key*, then Key  
180 Material is a structure as follows:

Object	Encoding	Required Field
Key Material	Structure	Yes
Modulus	Big Integer	Yes
Public Exponent	Big Integer	Yes

### 181 **2.1.7.6 Transparent DH Private Key**

182 If the Key Value Type in the Key Block is *Transparent DH Private Key*, then Key  
183 Material is a structure as follows:

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

184 Note:  $Q=P-1$ ,  $J$  where  $P=JQ+1$ .

### 185 **2.1.7.7 Transparent DH Public Key**

186 If the Key Value Type in the Key Block is *Transparent DH Public Key*, then Key  
187 Material is a structure as follows:

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

188  $Y=G^X \text{ mod } P$ ,  $Q=P-1$ ,  $J$  where  $P=JQ+1$ .

### 189 **2.1.7.8 Transparent ECDSA Private Key**

190 If the Key Value Type in the Key Block is *Transparent ECDSA Private Key*, then  
191 Key Material is a structure as follows:

Object	Encoding	Required Field
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
D	Big Integer	Yes

192

### 2.1.7.9 Transparent ECDSA Public Key

193

If the Key Value Type in the Key Block is *Transparent ECDSA Public Key*, then Key Material is a structure as follows:

194

Object	Encoding	Required Field
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
Q String	Octet String	Yes

195

### 2.1.7.10 Transparent ECDH Private Key

196

If the Key Value Type in the Key Block is *Transparent ECDH Private Key*, then Key Material is a structure as follows:

197

Object	Encoding	Required Field
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
D	Big Integer	Yes

198

### 2.1.7.11 Transparent ECDH Public Key

199

If the Key Value Type in the Key Block is *Transparent ECDH Public Key*, then Key Material is a structure as follows:

200

Object	Encoding	Required Field
Key Material	Structure	Yes
Recommended Curve	Enumeration	Yes
Q String	Octet String	Yes

201

## 2.1.8 Template-Attribute Structures

202

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

203

204

205

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

206



Object	Encoding	Required Field
Template-Attribute, Common Template-Attribute, Private Key Template- Attribute, Public Key Template-Attribute	Structure	Yes
Template Name	Text String	No, May be repeated.
Attribute	Attribute Object, see Section 2.1.1	No, May be repeated

The Template Name is the Name of a Template object or a Policy Template object, as defined in Sections 2.2.6 and 2.2.7 .

## 207 2.2 Managed Objects

208 Managed Objects are objects that are the subjects of key management operations, which are  
209 described in Sections 4 and 5 Managed Objects include all objects that may be registered with  
210 the system. *Managed Cryptographic Objects* are the subset of Managed Objects that contain  
211 cryptographic material, e.g. certificates, keys, and secret data. Managed Cryptographic Objects  
212 may have operations performed on them, and may have attributes that do not apply to all  
213 Managed Objects.

### 214 2.2.1 Certificate

215 A Managed Cryptographic Object, which is a digital certificate, such as an encoded X.509  
216 certificate.

Object	Encoding	Required Field
Certificate	Structure	Yes
Certificate Type	Enumeration	Yes
Certificate Value	Octet String	Yes

### 217 2.2.2 Symmetric Key

218 A Managed Cryptographic Object, which is a symmetric key.

Object	Encoding	Required Field
Symmetric Key	Structure	Yes
Key Block	Structure	Yes

### 219 2.2.3 Public Key

220 A Managed Cryptographic Object, which is the public portion of an asymmetric key pair.  
221 This is a “raw” public key, not a certificate.

Object	Encoding	Required Field
Public Key	Structure	Yes
Key Block	Structure	Yes

### 222 2.2.4 Private Key

223 A Managed Cryptographic Object, which is a the private portion of an asymmetric key  
224 pair.

Object	Encoding	Required Field
Private Key	Structure	Yes
Key Block	Structure	Yes

225

## 2.2.5 Split Key

226

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

227

228

229

230

231

232

Object	Encoding	Required Field
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, required only if Split Key Method is Polynomial Sharing Prime Field.
Key Block	Structure	Yes

233

There are three *Split Key Methods* for secret sharing: the first one is based on XOR and the other two are based on polynomial secret sharing, according to [Adi Shamir, "How to share a secret", Communications of the ACM, vol. 22, no. 11, pp. 612-613], as explained further in the Usage Guide. Let  $L$  be the minimum number of bits needed to represent all values of the secret.

234

235

236

237

238

- When the Split Key Method is XOR, the Key Material in the Key Value of the Key Block is of length  $L$  bits. The number of split keys is Split Key Parts (identical to Split Key Threshold), and the secret is reconstructed by XOR'ing all of them

239

240

241

- When the Split Key Method is Polynomial Sharing Prime Field, secret sharing is performed in the field  $GF(Prime\ Field\ Size)$ , represented as integers, where Prime Field Size is a prime bigger than  $2^L$ .

242

243

244

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

245

246

247

248

249

250

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

251

252

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

253

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

254

255

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

256

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

258 An element of  $GF(2^8)$  is represented as an octet. Addition and subtraction in  
 259  $GF(2^8)$  can be performed as a bitwise XOR of the octets. Multiplication and  
 260 inversion are more complex: see FIPS PUB 197 Section 4.1 and 4.2 for details.

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

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

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

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

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

269 The inverse of an element  $uy + v$  is given by

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

## 271 2.2.6 Template

272 A Template is a named Managed Object containing the client-settable attributes of a  
 273 Managed Cryptographic Object. It is essentially a stored, named list of attributes. A  
 274 Template is used to specify the attributes of a new Managed Cryptographic Object in  
 275 various operations. It is intended to be used to specify the cryptographic attributes of new  
 276 objects in a standardized or convenient way. None of the attributes specified in a  
 277 Template except the Name attribute apply to the template object itself, but instead apply  
 278 to any object created or registered using the Template.

279 The Template may be the subject of the Register, Locate, Get, Get Attributes, Get  
 280 Attribute List, Add Attribute, Modify Attribute, Delete Attribute, and Destroy operations.

281 The Template must have the Unique Identifier and Name attributes. They are applicable  
 282 to the Template itself, not to objects to which it is applied.

283 The attributes that may be contained in a Template are:

- 284 • Cryptographic Algorithm
- 285 • Cryptographic Length
- 286 • Cryptographic Parameters
- 287 • Object Group
- 288 • Application Specific Identification
- 289 • Contact Information
- 290 • Custom Attribute

Object	Encoding	Required Field
Template	Structure	Yes
Attribute	Attribute Object, see Section 2.1.1	Yes. May be repeated.

## 291 2.2.7 Policy Template

292 A *Policy Template* is a named Managed Object containing attributes. The purpose of a  
 293 Policy Template is to encapsulate all of the policy-related attributes into a Managed  
 294 Object which may be independent of any single Managed Cryptographic Object, and may

295 be managed and transmitted independently. Only policy-related attributes may be stored  
 296 in a Policy Template. The Policy Template may be the subject of the Register, Locate,  
 297 Get, get Attributes, Get Attribute List, Add Attribute, Modify Attribute, Delete Attribute, and  
 298 Destroy operations.

299 The Policy Template must have the Unique Identifier and Name attributes. They are  
 300 applicable to the Template itself, not to objects to which it is applied.

301 The attributes that may be contained in a Policy Template are:

- 302 • Cryptographic Algorithm
- 303 • Cryptographic Parameters
- 304 • Operation Policy Name
- 305 • Cryptographic Usage Mask
- 306 • Usage Limits
- 307 • Activation Date
- 308 • Process Start Date
- 309 • Protect Stop Date
- 310 • Deactivation Date
- 311 • Custom Attribute

Object	Encoding	Required Field
Policy Template	Structure	Yes
Attribute	Attribute Object, see Section 2.1.1	Yes. May be repeated

### 312 2.2.8 Secret Data

313 A Managed Cryptographic Object containing a shared secret that is not a key or  
 314 certificate, e.g., a password. The Key Block used to contain *Secret Data* should contain a  
 315 (possibly wrapped) Key Value of the Opaque type.

Object	Encoding	Required Field
Secret Data	Structure	Yes
Secret Data Type	Enumeration	Yes
Key Block	Structure	Yes

### 316 2.2.9 Opaque Object

317 A Managed Object that the key management server may not be able to interpret, but will  
 318 store. The context information for this object can be stored and retrieved using Custom  
 319 Attributes.

Object	Encoding	Required Field
Opaque Object	Structure	Yes
Opaque Data Type	Enumeration	Yes
Opaque Data Value	Octet String	Yes

## 320 3 Attributes

321 The following subsections describe the attributes that are associated with Managed Objects. These  
322 attributes may be obtained by a client from the server using the Get Attribute operation. Some attributes  
323 may be set by the Add Attribute operation or updated by the Modify Attribute operation, and some may be  
324 deleted by the Delete Attribute operation if they no longer apply to the Managed Object.

325 When attributes are returned by the server, e.g. via a Get Attributes operation, the returned attribute value  
326 may differ depending on the client. For example, the Cryptographic Usage Mask value may be different  
327 for different clients, depending on the policy of the server. Similarly, when a client modifies an attribute,  
328 this is merely a mechanism for sending information to the server. The server may store the attribute as  
329 received, or modify the attribute before saving it, or combine it with information from other sources, or  
330 merely use it as advice on how to modify its internal knowledge of the cryptographic object. The choice  
331 depends on server functionality, policy, and the kind of attribute being modified.

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

335 The second table in each subsection lists certain attribute characteristics, such as “Must always have a  
336 value”. The “When implicitly set” characteristic indicates which operations (other than operations that  
337 manage attributes) can implicitly result in adding or modifying the attribute of the object. They can be  
338 object(s) on which the operation is performed or object(s) created as a result of the operation. Implicit  
339 attribute changes occur even if the attribute is not specified in the operation request itself.

### 340 3.1 Unique Identifier

341 The *Unique Identifier* is generated by the key management system to uniquely identify a  
342 Managed Object. It is only required to be unique within the identifier space managed by a single  
343 key management system, however it is recommended that this identifier be globally unique, to  
344 allow for key management domain export of such objects. This attribute is assigned by the key  
345 management system at creation or registration time, and may never be changed or deleted by  
346 any entity at any time.

Object	Encoding	Required Field
Unique Identifier	Text String	Yes

347

Must 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

### 348 3.2 Name

349 The *Name* attribute is used to identify and locate the object, assigned by the client. The key  
350 management system may specify rules for valid names which may be created by the client.

351 Clients will be informed of such rules by a mechanism which is not specified here. Names must  
 352 be unique within a given key management domain, but are not required to be globally unique.

Object	Encoding	Required Field
Name	Structure	Yes
Name Value	Text String	Yes
Name Type	Enumeration	Yes

353

Must 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

### 354 3.3 Object Type

355 The type of a Managed Object, e.g. public key, private key, symmetric key, etc. This attribute is  
 356 set by the server when the object is created or registered and is never changed.

Object	Encoding	Required Field
Object Type	Enumeration	Yes

357

Must 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

### 358 3.4 Cryptographic Algorithm

359 The cryptographic algorithm used by the object, e.g. RSA, DSA, DES, 3DES, AES, etc. This  
 360 attribute is set by the server when the object is created or registered and is never changed.

Object	Encoding	Required Field
Cryptographic Algorithm	Enumeration	Yes

361

Must 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	All Cryptographic Objects

362

### 3.5 Cryptographic Length

363

*Cryptographic Length* is the length in bits of the cryptographic key material of the Managed Cryptographic Object. This attribute is set by the server when the object is created or registered, and is never changed.

364

365

Object	Encoding	Required Field
Cryptographic Length	Integer	Yes

366

Must 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	All Cryptographic Objects

367

### 3.6 Cryptographic Parameters

368

The *Cryptographic Parameters* attribute is a structure that contains a set of optional fields that describe certain cryptographic parameters to be used when performing cryptographic operations using the object. Specific fields may only pertain to certain types of Managed Cryptographic Objects.

369

370

371

Object	Encoding	Required Field
Cryptographic Parameters	Structure	Yes
Block Cipher Mode	Enumeration	No
Padding Method	Enumeration	No
Hashing Algorithm	Enumeration	No
Role Type	Enumeration	No

Must 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	All Cryptographic Objects

372 Role Types are defined as follows:

ZMK – Shared key to allow transfer of subordinate keys between two entities
ZPK – Shared key to allow transfer of PINs between two entities
MAC – MAC key, specifically X9.9/19 retail MAC
CVK – Key for generating/verifying 3-digit VISA/Mastercard signature strip codes (CVV/CVC)
CSC – Key for generating/verifying 4-digit American Express Card Security Codes
PVKIBM – Derivation key for derived PINs checked with the IBM offset method
PVKPVV – Verification key for random PINs checked with the PVV method
MKCVC – Master key for dynamic CVC calculations
MKSMI – Master key for smart card secure messaging integrity
MKSMC – Master key for smart card secure messaging confidentiality
MKIDN – Master key for Card Dynamic Number
MKAC – Master key for Chip card cryptogram
MKCAP – Master key for Cardholder Authentication Programme
BDK – Base derivation key for DUKPT

### 373 3.7 Certificate Type

374 The type of a certificate, e.g. X.509, PGP, etc. This value is set by the server when the certificate  
375 is created or registered and is never changed.

Object	Encoding	Required Field
Certificate Type	Enumeration	Yes

376



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

377

### 3.8 Certificate Issuer

378

An identification of a certificate, containing the Issuer Distinguished Name (from the Issuer field of the certificate) and the Certificate Serial Number (from the Serial Number field of the certificate).

379

This value is set by the server when the certificate is created or registered and is never changed.

380

Object	Encoding	Required Field
Certificate Issuer	Structure	Yes
Issuer	Text String	Yes
Serial Number	Text String	Yes (for X.509 certificates) / No (for PGP certificates since they don't contain a serial number)

381

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

382

### 3.9 Certificate Subject

383

Identifies the subject of a certificate, containing the Subject Distinguished Name (from the Subject field of the certificate). It may optionally include one or more alternative names (e.g. email address, IP address, DNS name) for the subject of the certificate (from the Subject Alternative Name extension within the certificate). These values are set by the server when the certificate is created or registered and are not changed until the certificate is renewed.

384

385

386

387

388

It is possible to issue an X.509 certificate where the subject field is left blank as long as the Subject Alternative Name extension is included in the certificate and is marked *CRITICAL*.

389

Therefore an empty string is an acceptable value for Certificate Subject.

390

Object	Encoding	Required Field
Certificate Subject	Structure	Yes

Certificate Subject Distinguished Name	Text String	Yes
Certificate Subject Alternative Name	Text String	No, May be repeated

391

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

392

### 3.10 Digest

393

394

395

396

397

A digest of the key or secret data (digest of the Key Material), certificate (digest of the Certificate Value), or opaque object (digest of the Opaque Data Value). Multiple digests may be calculated using different algorithms. The mandatory digest is computed with the SHA-256 hashing algorithm, the server can store additional optional digests. The digest(s) are static and generated by the server when the object is created or registered.

Object	Encoding	Required Field
Digest	Structure	Yes
Hashing Algorithm	Enumeration	Yes
Digest Value	Octet String	Yes

398

Must 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

399

### 3.11 Operation Policy Name

400

401

402

403

404

An indication of what entities may perform which key management operations on the object. The contents of the *Operation Policy Name* attribute is the name of a policy object known to the key management system and therefore server dependent. The named policy objects are created and managed using mechanisms outside the scope of the protocol. The policies determine who may perform specified operations on the object, and which of the objects' attributes may be modified,

405 or deleted, and by whom. It is expected that the Operation Policy Name attribute will be set when  
 406 operations such as Create or Register are executed. It is set either explicitly or via some default  
 407 set by the server, and will then apply to all subsequent operations on the object.

Object	Encoding	Required Field
Operation Policy Name	Text String	Yes

408

Must 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

### 409 3.11.1 Operations outside of operation policy control

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

- 412 • Create
- 413 • Create Key Pair
- 414 • Register
- 415 • Certify
- 416 • Validate
- 417 • Query
- 418 • Cancel
- 419 • Poll

### 420 3.11.2 Default Operation Policy

421 A key management system implementation should implement at least one named  
 422 operation policy, which is used for objects where the *Operation Policy* attribute is not  
 423 specified by the Client in a *Create* or *Register* operation, or in a template specified in  
 424 these operations. This policy is named *default*. It specifies the following rules for  
 425 operations on objects created or registered with this policy, depending on the object type.

#### 426 3.11.2.1 Default Operation Policy for Secret Objects

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

<b>Default Operation Policy for Secret Objects</b>	
<b>Operation</b>	<b>Policy</b>
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

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

431 **3.11.2.2 Default Operation Policy for Certificates and Public**  
432 **Key Objects**

433 This policy applies to Certificates and Public Keys.

<b>Default Operation Policy for Certificates and Public Key Objects</b>	
<b>Operation</b>	<b>Policy</b>
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

### 3.11.2.3 Default Operation Policy for Template and Policy Template Objects

The operation policy specified as an attribute in the *Create* operation for a template object is the operation policy used for objects that will be created using that template, and is not the policy used to control operations on the template itself. There is no mechanism provided for specifying a policy used to control operations on template objects, so the default policy for template objects themselves 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

In addition to private template objects, which are controlled by the above policy which can be created by clients or the server, publicly known and usable templates may be created and managed by the server, with a different default policy for these 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

447

### 3.12 Cryptographic Usage Mask

448  
449

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

450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463

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

464  
465  
466

This list takes into consideration values which may appear in the Key Usage extension in an X.509 certificate. However, the list does not consider the more fined grained usages which may appear in the Extended Key Usage extension.

467  
468

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

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

469  
470  
471  
472  
473

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

Object	Encoding	Required Field
Cryptographic Usage Mask	Integer	Yes

474

Must 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

475

### 3.13 Lease Time

476

477

478

479

480

481

The *Lease Time* attribute defines a time interval for a Managed Object that indicates how long a client should use the object. This attribute always holds the initial value of a lease, and not the actual remaining time. Note that once the lease expires, the client must renew the lease by calling Obtain Lease. A server should store in this attribute the maximum Lease Time it is willing to serve and a client must request lease times (with Obtain Lease) which are less than, or equal. This attribute is read-only for clients. It can be modified by the server only.

Object	Encoding	Required Field
Lease Time	Interval	Yes

482

Must 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 Keys

483

### 3.14 Usage Limits

484

485

486

487

488

489

490

491

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

- 492 • *Usage Limits Total Bytes* – the total number of bytes allowed to be protected. This is the  
493 total value for the entire life of the object, and is never changed once the object begins to  
494 be used for protection purposes.
- 495 • *Usage Limits Total Objects* – the total number of objects allowed to be protected. This is  
496 the total value for the entire life of the object, and is never changed once the object  
497 begins to be used for protection purposes.
- 498 • *Usage Limits Byte Count* – the currently remaining number of bytes allowed to be  
499 protected.
- 500 • *Usage Limits Object Count* – the currently remaining number of objects allowed to be  
501 protected.

502 When the attribute is initially set, usually during object creation or registration, the values set are  
503 the Total values allowed for the useful life of the object. The count values must be ignored by the  
504 server if the attribute is specified in a operation that creates a new object. Changes made via the  
505 Modify Attribute operation reflect corrections to these Total values, but they cannot be changed  
506 once the count values have changed by a Get Usage Allocation operation. The count values  
507 cannot be set or modified by the client via the Add Attribute or Modify Attribute operations.

Object	Encoding	Required Field
Usage Limits	Structure	Yes
Usage Limits Total Bytes	Big Integer	No. Must be present if Usage Limits Byte Count is present
Usage Limits Total Objects	Big Integer	No. Must be present if Usage Limits Object Count is present
Usage Limits Byte Count	Big Integer	No. May only be present if Usage Limits Object Count is not present
Usage Limits Object Count	Big Integer	No. May only be present if Usage Limits Byte Count is not present

508

Must 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	Symmetric Keys, Private Keys, Split Keys, Public Keys



509

### 3.15 State

510  
511  
512  
513  
514

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

515  
516

- *Pre-Active*: The object exists but is not yet usable for any cryptographic purpose.

517  
518  
519

- *Active*: The object may be used for all cryptographic purposes which are allowed by its Cryptographic Usage Mask attribute.

520  
521  
522  
523  
524  
525  
526  
527

- *Deactivated*: The object may not be used for protection purpose, e.g. encryption or signing, but, if permitted by the Cryptographic Usage Mask attribute, may be used for process purposes, e.g. decryption or verification, but only under extraordinary circumstances and when special permission is granted.

528  
529  
530  
531

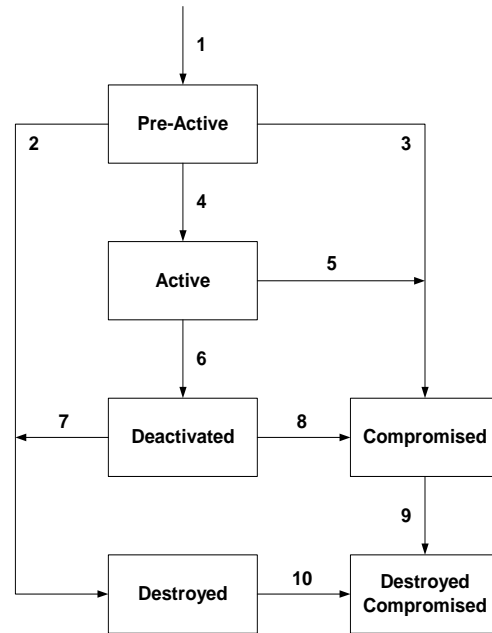
- *Compromised*: The object may have been compromised, and may only be used for process purposes in a client that is trusted to handle compromised cryptographic objects.

532  
533

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

534  
535  
536

- *Destroyed Compromised*: The object is no longer usable for any purpose, however its compromised status may be retained for audit or security purposes.



537

State transitions occur as follows:

538  
539  
540  
541  
542  
543  
544  
545

1. The transition from a non-existent key to Pre-Active is determined by the creation of the object. When an object is created or registered, it automatically goes from non-existent to Pre-Active. If, however, the operation that creates or registers the object contains an Activation Date that has already occurred, the state immediately transitions to Active. In this case, the server may set the Activation Date attribute to the time when the operation is received, depending on server policy. If the operation contains an Activation Date attribute in the future, or contains no Activation Date, it becomes initialized in the key management system in the Pre-Active state.

546  
547

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

548

3. The transition from Pre-Active to Active can occur in one of two ways:

549  
550

- The object has an Activation Date in the future. At the time the Activation Date is reached, the server may change the state to Active.

551  
552  
553  
554

- A client issues a Modify Attribute operation, modifying the Activation Date to a date in the past, or the current date. In this case, the server may set the Activation Date attribute to the time when the operation that created or registered the object was received, depending on server policy.

555  
556

- A client issues an Activate operation on the object. The server will set the Activation Date to the time the Activate operation is received.

- 557 4. The transition from Active to Compromised is performed by a client issuing a Revoke  
558 operation with a Revocation Reason of Compromised.
- 559 5. The transition from Active to Deactivated can occur in one of two ways:
- 560 • The object's Deactivation Date is reached. The server may change the state to  
561 Deactivated.
  - 562 • A client issues a Revoke operation, with a Revocation Reason other than Compromised.
  - 563 • The client issues a Modify Attribute operation, modifying the Deactivation Date to a date in  
564 the past, or the current date. In this case, the server may set the Deactivation Date attribute  
565 to the date in the past or the current date, depending on server policy.
- 566 6. The transition from Deactivated to Destroyed is requested by a client issuing a Destroy  
567 operation. The server will destroy the object when and if server policy dictates.
- 568 7. The transition from Deactivated to Compromised is performed by a client issuing a Revoke  
569 operation with a Revocation Reason of Compromised.
- 570 8. The transition from Compromised to Destroyed Compromised is requested by a client issuing  
571 a Destroy operation. The server will destroy the object when and if server policy dictates.
- 572 9. The transition from Destroyed to Destroyed Compromised is performed by a client issuing a  
573 Revoke operation with a Revocation Reason of Compromised.
- 574 Only the transitions described above are permitted.

Object	Encoding	Required Field
State	Enumeration	Yes

575

Must 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

### 576 3.16 Initial Date

577 The date and time when the Managed Object was first created or registered at the server. This  
578 time corresponds to state transition 1 (see Section 3.15 ). This attribute is set by the server when  
579 the object is created or registered, and is never changed. This attribute is also set for non-  
580 cryptographic objects (e.g. templates) when then are first registered with the server.

Object	Encoding	Required Field
Initial Date	Date-Time	Yes

581

Must 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

582 **3.17 Activation Date**

583 The date and time when the Managed Cryptographic Object may begin to be used. This time  
584 corresponds to state transition 4 (see Section 3.15 ). The object may not be used for any  
585 cryptographic purpose before the *Activation Date* has been reached. Once the state transition has  
586 occurred, this attribute may no longer be modified by the server or client. If a client attempts to set  
587 this value to a time in the past, the server may set it to the current time instead, depending on  
588 server policy.

Object	Encoding	Required Field
Activation Date	Date-Time	Yes

589

Must 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

590 **3.18 Process Start Date**

591 The date and time when a Managed Symmetric Key Object may begin to be used for process  
592 purposes, e.g. decryption or unwrapping, depending on the value of its Cryptographic Usage  
593 Mask attribute. The object may not be used for these cryptographic purposes before the *Process  
594 Start Date* has been reached. This value may be equal to, but may not precede, *Activation Date*.  
595 Once the Process Start Date has occurred, this attribute may no longer be modified by the server  
596 or the client. If a client attempts to set this value to a time in the past, the server may set it to the  
597 current time instead, depending on server policy.

Object	Encoding	Required Field
Process Start Date	Date-Time	Yes

Must 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, Re-key
Applies to Object Types	Symmetric Keys and Split Keys of symmetric keys

599 **3.19 Protect Stop Date**

600 The date and time when a Managed Symmetric Key Object may no longer be used for protect  
 601 purposes, e.g. encryption or wrapping, depending on the value of its Cryptographic Usage Mask  
 602 attribute. This value may be equal to, but may not be later than Deactivation Date. Once the  
 603 *Protect Stop Date* has occurred, this attribute may no longer be modified by the server or the  
 604 client. If a client attempts to set this value to a time in the past, the server may set it to the current  
 605 time instead, depending on server policy.

Object	Encoding	Required Field
Protect Stop Date	Date-Time	Yes

606

Must 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, Re-key
Applies to Object Types	Symmetric Keys and Split Keys of symmetric keys

607 **3.20 Deactivation Date**

608 The date and time when the Managed Cryptographic Object may no longer be used for any  
 609 purpose, except for decryption, signature verification, or unwrapping, but only under extraordinary  
 610 circumstances and when special permission is granted. This time corresponds to state transition  
 611 6 (see Section 3.15 ). Once this transition has occurred, this attribute may no longer be modified  
 612 by the server or client. If a client attempts to set this value to a time in the past, the server may set  
 613 it to the current time instead, depending on server policy.

Object	Encoding	Required Field
Deactivation Date	Date-Time	Yes

614

Must 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

615 **3.21 Destroy Date**

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

Object	Encoding	Required Field
Destroy Date	Date-Time	Yes

620

Must 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 Objects

621 **3.22 Compromise Occurrence Date**

622 The date and time when the Managed Cryptographic Object was first believed to be  
623 compromised. If it is not possible to estimate when the compromise occurred, this value should  
624 be set to the Initial Date for the object.

Object	Encoding	Required Field
Compromise Occurrence Date	Date-Time	Yes

625

Must 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	Symmetric Keys, Private Keys, Split Keys, Secret Data, Opaque Object, Certificates

626 **3.23 Compromise Date**

627 The date and time when the Managed Cryptographic Object is entered into the compromised  
628 state. This time corresponds to state transitions 3, 5, 8, or 10 (see Section 3.15 ). This time  
629 represents when the key management system was made aware of the compromise, not  
630 necessarily when the compromise occurred. This attribute is set by the server when it receives a  
631 Revoke operation with a Revocation Reason of Compromised.

Object	Encoding	Required Field
Compromise Date	Date-Time	Yes

632

Must 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	Symmetric Keys, Private Keys, Split Keys, Secret Data, Opaque Object, Certificates

633 **3.24 Revocation Reason**

634 An indication of why the Managed Cryptographic Object was revoked, e.g. “compromised”,  
635 “expired”, “no longer used”, etc. This attribute is only changed by the server as a side effect of the  
636 Revoke Operation.

637 The *Revocation Message* is an optional field which is used exclusively for audit trail/logging  
638 purposes and may contain additional information about why the object was revoked, for example  
639 “Laptop stolen”, or “Machine decommissioned”.

Object	Encoding	Required Field
Revocation Reason	Structure	Yes
Revocation Reason Code	Enumeration	Yes
Revocation Message	Text String	No

640

Must 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

641

### 3.25 Archive Date

642

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

643

644

Object	Encoding	Required Field
Archive Date	Date-Time	Yes

645

Must 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

646

### 3.26 Object Group

647

An object may be part of a group of objects. An object may belong to more than one group. To assign an object to a group, the group name should be set into this attribute. The key management system may specify rules for the valid group names which may be created by the client. Clients will be informed of such rules by a mechanism which is not specified by this standard. In the protocol, the group names themselves are character strings of no specified format. Specific key management system implementations may choose to support hierarchical naming schemes or other syntax restrictions on the names. Groups may be used to associate objects for a variety of purposes. A set of keys used for a common purpose, but for different time intervals, may be linked by a common Object Group. Servers may create predefined groups and add objects to them independently of client requests.

648

649

650

651

652

653

654

655

656

Object	Encoding	Required Field
Object Group	Text String	Yes

657

Must 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

658

### 3.27 Link

659

A link from a Managed Cryptographic Object to another, closely related target Managed Cryptographic Object. The link has a type and the allowed types differ depending on the Object Type of the Managed Cryptographic Object. The *Linked Object Identifier* identifies the target Managed Cryptographic Object by its Unique Identifier. The link can contain such information as the private key corresponding to a public key, the parent certificate for a certificate in a chain, or for a derived symmetric key, the base key from which it was derived.

660

661

662

663

664

665

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

666

667

- *Private Key Link*. For a Public Key object: the private key corresponding to the public key

668

- *Public Key Link*. For a Private Key object: the public key corresponding to the private key. For a Certificate object: the public key certified by the certificate

669

670

- *Certificate Link*. For Certificate objects: the parent certificate for a certificate in a certificate chain. For Public Key objects: the corresponding certificate(s), containing the same public key

671

672

673

- *Derivation Base Object Link* for a derived Symmetric Key object: the object(s) from which the current symmetric key was derived

674

675

- *Derived Key Link*: the symmetric key(s) that were derived from the current object.

676

- *Replacement Object Link*. For a Symmetric Key, Private Key, or Public Key object: the key that resulted from the re-key of the current key. For a Certificate object: the certificate that resulted from the re-certify. Note there can only be one such replacement object.

677

678

679

- *Replaced Object Link*. For a Symmetric Key, Private Key, or Public Key object: the key that was re-keyed to obtain the current key. For a Certificate object: the certificate that was re-certified to obtain the current certificate

680

681

682

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

683

684

Note that a Managed Object may have a Link attribute which has multiple values. For example, a Private Key may have links to the associated certificate as well as the associated public key. As another example, a Certificate object may have a Link attribute value to both the public key and to the certificate of the certification authority which signed the certificate.

685

686

687



688 It is also possible that a Managed Object does not have Link attribute values for associated  
 689 cryptographic objects. This can occur in cases where the associated key material is not available  
 690 to the server or client (consider the registration of a CA Signer certificate with a server but the  
 691 corresponding private key is held in a different manner).

Object	Encoding	Required Field
Link	Structure	Yes
Link Type	Enumeration	Yes
Linked Object Identifier	Text String	Yes

692

Must always have a value	No
Initially set by	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 Objects

### 693 3.28 Application Specific Identification

694 The *Application Specific Identification* is used to specify the intended use of a Managed Object. It  
 695 consists two parts: the application name space that the object will be used with, and an  
 696 identification specific to that application name space. The application name spaces are arbitrary  
 697 text strings so that new types of application identifiers can be used without requiring the standard  
 698 to be updated.

699 Some examples of application name space and identifier pairs:

- 700 • SMIME, 'someuser@company.com'
- 701 • SSL, 'some.domain.name'
- 702 • Volume Identification, '123343434'
- 703 • File Name, 'secret.doc'

704 The following application names spaces are recommended:

- 705 • SMIME
- 706 • SSL
- 707 • IPSEC
- 708 • HTTPS
- 709 • PGP
- 710 • Volume Identification
- 711 • File Name

712 Other values may be used according to server policy. No extension mechanism is defined or  
 713 needed as any text string is allowable.

Object	Encoding	Required Field
Application Specific Identification	Structure	Yes
Application Name Space	Text String	Yes
Application Identifier	Text String	Yes

714

Must 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	All Cryptographic Objects

715

### 3.29 Contact Information

716

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

717

Object	Encoding	Required Field
Contact Information	Text String	Yes

718

Must 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

719

### 3.30 Last Changed Date

720

A meta attribute that contains the date and time of the last change to the contents or attributes of the specified object.

721

Object	Encoding	Required Field
Last Changed Date	Date-Time	Yes

722

Must 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, Activate, Revoke, Destroy, Archive, Recover, Certify, Re-certify, Re-key, Get Usage Allocation
Applies to Object Types	All Objects

723

### 3.31 Custom Attribute

724

725

726

727

728

729

730

A *Custom Attribute* is user-defined attribute and intended for vendor-specific purposes. It is created by the client and not interpreted by the server, or created by the server and either understood or not understood by the client. All custom attributes created by the client must adhere to a naming scheme where the name of the attribute must have a prefix of 'x-', meaning extended. The key management server may create and manage custom attributes which have a prefix of 'y-'. The tag type Custom Attribute cannot identify the particular attribute, hence such an attribute can only appear in an Attribute Structure with its name as defined in Section 2.1.1 .

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

731

Must 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

732

## 4 Client-to-Server Operations

733

734

735

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 must be capable of understanding the response to the request. All Object Management

736 operations are sent in requests from clients to servers, and in responses from servers to clients. These  
737 operations may be combined into a batch, which allows multiple operations to be contained in a single  
738 request/response message pair.

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

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

746 The ID Placeholder is obtained from the Unique Identifier returned by the Create, Create Pair, Register,  
747 Derive Key, Re-Key, Certify, Re-Certify, Locate, and Recover operations. If any of these operations  
748 successfully completes and returns a Unique Identifier, then the server must copy this Unique Identifier  
749 into the ID Placeholder variable, where it is held until the completion of the operations remaining in the  
750 batched request. Subsequent operations in the batched request that need a Unique Identifier may make  
751 use of the ID Placeholder. This is indicated by omitting the Unique Identifier field from the request  
752 payloads for these operations. This mechanism is only valid if the Batch Error Continuation Option is set  
753 to Stop and the Batch Order Option is set to true.

754 Requests may contain attribute values to be assigned to the object. This information is specified with a  
755 Template-Attribute (see Section 2.1.8 ) that contains zero or more template names and zero or more  
756 individual attributes. If more than one template is specified, and there is a conflict between the single-  
757 value attributes in the templates, the value in the subsequent template takes precedence. If there is a  
758 conflict between the single-value attributes in the request and the single-value attributes in a specified  
759 template, the attribute values in the request take precedence. For multi-value attributes, the union of  
760 attribute values is used when the attributes are specified more than once.

761 Responses may contain attribute values that have been set differently than specified in the request. This  
762 information is specified with a Template-Attribute that contains one or more individual attributes.

## 763 **4.1 Create**

764 This operation requests the server to generate a new key as a Managed Cryptographic Object.  
765 This operation is not used to create Template or Policy Template objects (see Register operation,  
766 Section 4.3 ).

767 The request contains information about the type of object being created, and some of the  
768 attributes to be assigned to the object, e.g. Cryptographic Algorithm, Cryptographic Length, etc.  
769 This information may be specified by the names of Template objects which already exist. The  
770 response contains the Unique Identifier of the created object. The server must copy the Unique  
771 Identifier returned by this operation into the ID Placeholder variable.

Request Payload		
Object	Required Field	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

772

Response Payload		
Object	Required Field	Description
Object Type	Yes	Type of object created
Unique Identifier	Yes	The Unique Identifier of the newly created object
Template-Attribute	No	A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here

773 The following attributes must be included in the Create request, either explicitly, or via  
774 specification of a template that contains the attribute.

Attribute	Required
Cryptographic Algorithm	Yes
Cryptographic Usage Mask	Yes

## 775 4.2 Create Key Pair

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

778 The request contains attributes to be assigned to the objects, e.g. Cryptographic Algorithm,  
779 Cryptographic Length, etc. Attributes and Template Names can be specified for both keys at the  
780 same time, by specifying a Common Template-Attribute object in the request. Attributes not  
781 common to both keys (e.g., Name, Cryptographic Usage Mask) may be specified using the  
782 Private Key Template-Attribute and Public Key Template-Attribute objects in the request which  
783 take precedence over the Common Template-Attribute object. A Link Attribute is automatically  
784 created by the server for each object, pointing to the corresponding object. The response  
785 contains the Unique Identifiers of both created objects. The ID Placeholder value will be set to the  
786 Unique Identifier of the Private Key.

Request Payload		
Object	Required Field	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

787 For multi-valued attributes, the union of the values found in the templates and attributes of the  
788 Common, Private, and Public Key Template-Attribute is used. For single-valued attributes, the  
789 order of precedence is as follows:

- 790 1. attributes specified explicitly in the Private and Public Key Template-Attribute, then
- 791 2. attributes specified via templates in the Private and Public Key Template-Attribute, then
- 792 3. attributes specified explicitly in the Common Template-Attribute, then
- 793 4. attributes specified via templates in the Common Template-Attribute

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

Response Payload		
Object	Required Field	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	A list of attributes, for the Private Key Object, with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here
Public Key Template-Attribute	No	A list of attributes, for the Public Key Object, with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here

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

Attribute	Required	Must 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
Contact Information	No	Yes

798

### 4.3 Register

799

800

801

802

803

804

This operation requests the server to register a Managed Object (created by the client or obtained by the client through some other means), allowing the server to manage the object. The arguments in the request are similar to those in the Create operation, but also may contain the object itself, for storage by the server. Optionally, objects which the client does not wish to be stored by the key management system may be omitted from the request, for example, private keys.

805

806

807

808

809

810

The request contains information about the type of object being registered, and some of the attributes to be assigned to the object, e.g. Cryptographic Algorithm, Cryptographic Length, etc. This information may be specified by the use of a Template-Attribute object. The response contains the Unique Identifier assigned by the server to the registered object. The server must copy the Unique Identifier returned by this operations into the ID Placeholder variable. The Initial Date attribute of the object is set to the current time.

Request Payload		
Object	Required Field	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 cryptographic object being registered. The object and attributes may be wrapped. Some objects, e.g. Private Keys, may be omitted from the request

811

Response Payload		
Object	Required Field	Description
Object Type	Yes	Type of object registered
Unique Identifier	Yes	The Unique Identifier of the newly registered object
Template-Attribute	No	A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here

812 If the Register operation is being used to register a new Template, or Policy Template, then the  
813 request payload will contain a single Template Name field, containing the name of the new  
814 template, and the Cryptographic Object field must be omitted. The contents of the new Template  
815 or Policy Template will be the attributes contained in the Template-Attribute object in the request.

Request Payload		
Object	Required Field	Description
Object Type	Yes	Template or Policy Template
Template Name	Yes	Specifies the name of the Template being registered
Template-Attribute	Yes, May be repeated	Specifies the attributes of the new Template using templates and/or as individual attributes

816 When registering a new Template or a Policy Template, the attributes that may be included in the  
817 request are specified in Section 2.2.6 and 2.2.7 , respectively (note however that the Name  
818 attribute may not be specified). For all other object types that can be registered, the following  
819 attributes must be included in the Register request, either explicitly, or via specification of a  
820 template that contains the attribute.



Attribute	Required
Cryptographic Algorithm	Yes, may be omitted only if this information is encapsulated in the Key Block. Does not apply to Secret Data or Opaque Objects. If present, Cryptographic Length below must 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 or Opaque Objects. If present, Cryptographic Algorithm above must also be present.
Cryptographic Usage Mask	Yes

#### 821 4.4 Re-key

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

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

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

829 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
830 a Recover operation before they can be specified.

831 As a result of Re-key, attributes of the existing key are changed similarly to performing a Revoke  
832 on that key with a Revocation Reason of Superseded, and the Link attribute is set to point to the  
833 replacement key.

834 If Offset is set, then the times of the new key will be set based on the times of the existing key (if  
835 such times exist) 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 ( $DT_1 + (AT_2 - AT_1)$ )

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

<b>Attribute</b>	<b>Action</b>
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

837

<b>Request Payload</b>		
<b>Object</b>	<b>Required Field</b>	<b>Description</b>
Unique Identifier	No	Determines the Symmetric Key being re-keyed. If omitted, 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

838

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the new Symmetric Key
Template-Attribute	No	A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here

## 839 4.5 Derive Key

840 This request is used to derive a symmetric key using a key or secret that is already known to the  
841 key management system. It only applies to Managed Objects that can be used for key derivation  
842 (The Derive Key bit must be set in the Cryptographic Usage Mask attribute of the specified  
843 Managed Object). If the operation is issued for an object that does not have this bit set, the server  
844 must return a response with a Result Reason of Operation Not Supported. For all derivation  
845 methods, the client must specify the desired length of the derived key or secret using the  
846 Cryptographic Length attribute. If a key is created, the client must specify both the Cryptographic  
847 Length and Cryptographic Algorithm. If the specified length exceeds the output of the derivation  
848 method, the server must return an error. Clients have the option to derive multiple keys and IVs  
849 by creating a Secret Data object and specifying a Cryptographic Length that is the total length of  
850 the derived object. The length must not exceed the length of the output that is returned by the  
851 chosen derivation method.

852 The fields in the request specify the Unique Identifiers of the keys or secrets to be used for  
853 derivation (some derivation methods may require multiple keys or secrets to derive the result), the  
854 method to be used to perform the derivation, and any parameters needed by the specified  
855 method. The method is specified as an enumerated value. Currently defined derivation methods  
856 include:

- 857 • *PBKDF2* – This method is used to derive a symmetric key from a password or pass  
858 phrase. The PBKDF2 method is published in RSA Laboratories' Public-Key Cryptography  
859 Standards (PKCS) series, specifically PKCS #5 v2.0, and also published as Internet  
860 Engineering Task Force's RFC 2898.
- 861 • *HASH* – This method derives a key by computing a hash over the derivation key or the  
862 derivation data.
- 863 • *HMAC* – This method derives a key by computing an HMAC over the derivation data.
- 864 • *ENCRYPT* – This method derives a key by encrypting the derivation data.
- 865 • *NIST800-108-C* – This method derives a key by computing the KDF in Counter Mode as  
866 specified in NIST SP 800-108.
- 867 • *NIST800-108-F* – This method derives a key by computing the KDF in Feedback Mode  
868 as specified in NIST SP 800-108.
- 869 • *NIST800-108-DPI* – This method derives a key by computing the KDF in Double-Pipeline  
870 Iteration Mode as specified in NIST SP 800-108.
- 871 • *Extensions*

872 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
873 a Recover operation before they can be specified. The server must perform the derivation

874 function, and then register the derived object as a new Managed Object, returning the new  
 875 Unique Identifier for the new object in the response. The server must copy the Unique Identifier  
 876 returned by this operation into the ID Placeholder variable.

877 As a result of Derive Key, the Link attributes (Derived Key Link in the objects from which the key  
 878 is derived, and the Derivation Base Object Link in the derived key) of all objects involved are set  
 879 to point to the corresponding objects.

Request Payload		
Object	Required Field	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 from. At most two can be specified: one for the derivation key and another for the secret data. Note that the ID Placeholder cannot 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 must always be specified and algorithm is required for the creation of symmetric keys.

880

Response Payload		
Object	Required Field	Description
Object Type	Yes	Type of object created
Unique Identifier	Yes	The Unique Identifier of the newly derived key
Template-Attribute	No, May be repeated	A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here

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

Object	Encoding	Required Field
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

883 Cryptographic Parameters identify the Pseudorandom Function (PRF) or the mode of operation of  
884 the PRF. For example, if a key is derived using the HASH derivation method, clients are required  
885 to provide the hash algorithm inside Cryptographic Parameters. Similarly, if a key is derived using  
886 AES in CBC mode, clients are required to provide the Block Cipher Mode. The server will verify  
887 that the specified mode matches one of the instances of Cryptographic Parameters set for the  
888 corresponding key. If Cryptographic Parameters are omitted, the server will pick the  
889 Cryptographic Parameters set with the lowest index for the specified key. If the corresponding key  
890 does not have any Cryptographic Parameters attribute, or if no match is found, an error is  
891 returned.

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

894 Derivation Data can either be the data to be encrypted, hashed, or HMACed. For NIST SP 800-  
895 108 methods, Derivation Data is Label||{0x00}||Context, where the all-zero octet is optional.

896 Most derivation methods, such as ENCRYPT, require a derivation key and the derivation data to  
897 be encrypted. The HASH derivation method requires either a derivation key or derivation data.  
898 Derivation data can either be explicitly provided by the client with the Derivation Data field or  
899 implicitly by providing the Unique Identifier of a Secret Data object. An error is returned if both are  
900 provided.

901 The PBKDF2 derivation method requires two additional parameters:

Object	Encoding	Required Field
Derivation Parameters	Structure	Yes
Cryptographic Parameters	Structure	No, depends on the PRF
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

## 902 4.6 Certify

903 This request is used to obtain a new certificate for a public key. Only a single certificate can be  
904 requested at a time. Server support for this operation is optional, as it requires that the key  
905 management system have access to a certification authority.

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

908 The server must copy the Unique Identifier of the certificate returned by this operation into the ID  
 909 Placeholder variable. The new Certificate object whose Unique Identifier is returned may be  
 910 obtained by the client via a Get operation in the same batch, using the ID Placeholder  
 911 mechanism.

912 As a result of Certify, the Link attribute of the Public Key and of the new Certificate are set to point  
 913 at each other.

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

916 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 917 a Recover operation before they can be specified.

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

Request Payload		
Object	Required Field	Description
Unique Identifier	No	The Unique Identifier of the Public Key being certified. If omitted, 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

920

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the new certificate
Template-Attribute	No	A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here

## 921 **4.7 Re-certify**

922 This request is used to renew an existing certificate with the same key pair. Only a single  
 923 certificate can be renewed at a time. Server support for this operation is optional, as it requires  
 924 that the key management system have access to a certification authority.

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

927 The server must copy the Unique Identifier of the certificate returned by this operation into the ID  
 928 Placeholder variable.

929 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 930 a Recover operation before they can be specified.

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

933 As the new certificate takes over the name attribute of the existing certificate, Re-certify should  
 934 only be performed once on a given certificate.

935 As a result of Re-certify, attributes of the existing certificate are changed similarly to performing a  
 936 Revoke on that key with a Revocation Reason of Superseded.

937 In addition, the Link attribute of the existing certificate and of the new certificate are set to point at  
 938 each other. In addition, the Link attribute of the Public Key is changed to point to the new  
 939 certificate. If *Offset* is set, then the times of the new certificate will be set based on the times of  
 940 the existing certificate (if 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 ( $DT_1 + (AT_2 - AT_1)$ )

941 Attributes that are not copied from the existing certificate and 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

942

Request Payload		
Object	Required Field	Description
Unique Identifier	No	The Unique Identifier of the Certificate being renewed. If omitted, 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

943

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the new certificate
Template-Attribute	No	A list of object attributes with values that the key management server chose differently from those specified in the request (either explicitly or via template). Only those attributes that were specified in the request and were set to different values by the server are included here

944

## 4.8 Locate

945 This operation requests that the server searches for one or more Managed Objects, specified by  
946 one or more attributes. All attributes are allowed to be used. However, no attributes specified in  
947 the request should contain index values. Attribute Index values will be ignored by the *Locate*  
948 operation. The request may also contain a *Maximum Items* field, which specifies the maximum  
949 number of objects that the client wishes returned by *Locate*. If the *Maximum Items* field is omitted,  
950 then the server may return all objects matched, or may impose an internal maximum limit due to  
951 resource limitations.

952 The response may contain Unique Identifiers for multiple Managed Objects, if more than one  
953 object satisfies the identification criteria specified in the request. Returned objects must match **all**  
954 of the attributes in the request. If no objects match, an empty response payload is returned.

955 The server returns a list of Unique Identifiers of the found objects, which then must be retrieved  
956 using the *Get* operation, or if the objects are archived, then the *Recover* and *Get* operations must  
957 be used. The server must copy the Unique Identifier returned by this operation into the *ID*  
958 *Placeholder* variable. If the *Locate* operation matches more than one object, and the *Maximum*  
959 *Items* value is omitted in the request, or is set to a value larger than one, then the server must not



960 set the ID Placeholder value, so that any subsequent operations that are batched with the Locate,  
961 and which do not specify a Unique Identifier explicitly will fail. This ensures that these batched  
962 operations will be allowed to proceed only if a single object is returned by Locate.

963 When using the Name or Object Group attributes for identification, wild-cards or regular  
964 expressions may be supported by specific key management system implementations. The  
965 protocol neither requires nor disallows such use.

966 The Date attributes (Initial Date, Activation Date, etc) may be used to specify a time or a time  
967 range. If a single instance of a given Date attribute is used, such as Activation Date, then objects  
968 with the same Activation Date are matching candidate objects. If two instances of the same Date  
969 attribute are used (with two different values specifying a range), then objects for which the  
970 Activation Date is inside or on the range are matching candidate objects. If a Date attribute is set  
971 to its largest possible value, then it is equivalent to an undefined attribute.

972 When the Cryptographic Usage Mask attribute is specified in the request, candidate objects are  
973 matched against this field via an operation which consists of a logical AND of the requested mask  
974 with the mask in the candidate object and then a straight comparison of the resulting value with  
975 the requested mask. For example, if the request contains a mask value of 1000110001 and a  
976 candidate object mask contains 1000010001, the logical AND of the two masks is 1000010001  
977 which is compared against 1000110001 and fails the match. This means that a matching  
978 candidate object must have all of the bits set in its mask that are set in the requested mask, but  
979 may have additional bits set.

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

983 When an attribute defined as a structure is specified, not all of the structure fields must be  
984 specified. For instance, for the Link attribute, the Linked Object Identifier value may be specified  
985 without the Link Type value, and matching candidate objects must have the Linked Object  
986 Identifier as specified, irrespective of their Link Type.

987 The Storage Status Mask field (see Section 9.1.3.3.2 ) is used to indicate whether only on-line  
988 objects, or only archived objects, or both on-line and archived objects must be searched. Note  
989 that the server may store attributes of archived objects in order to expedite Locate operations  
990 searching through archived objects.

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

991

Response Payload		
Object	Required Field	Description
Unique Identifier	No, May be repeated	The Unique Identifier of the located objects

992 **4.9 Check**

993 This operation requests that the server checks for the use of a Managed Object according to  
 994 policy-related values specified in the request. This operation should only be used when placed in  
 995 a batched set of operations, usually following a Locate, Create, Create Pair, Derive Key, Certify,  
 996 Re-Certify or Re-Key operation and followed by a Get operation. The Unique Identifier field in the  
 997 request may be omitted if the operation is in a batched set of operations and follows an operation  
 998 that sets the ID Placeholder variable.

999 If the server determines that the client is allowed to use the object specified according to the  
 1000 given policy attributes, the server returns the Unique Identifier of the object. If the server  
 1001 determines that the specified attributes fall outside allowed policy, then the server returns no  
 1002 Unique Identifier, the server invalidates the ID Placeholder value, and the operation returns the  
 1003 set of attributes specified in the request that caused the server policy denial. Only those attributes  
 1004 that the server judged to be out of policy are returned, allowing the client to determine how to  
 1005 proceed. The operation also returns a failure, thus causing any subsequent operations in the  
 1006 batch to be ignored.

1007 The additional objects that may be specified in the request are limited to (note that these objects  
 1008 are not encoded in an Attribute structure as shown in Section 2.1.1 ):

- 1009 • Usage Limits Byte Count or Usage Limits Object Count (see Section 3.14 )– The request  
 1010 may contain the usage amount that the client deems necessary to complete its needed  
 1011 function. This does not require that any subsequent Get Usage Allocation operations  
 1012 request this amount. It only means that the client is ensuring that the amount specified is  
 1013 available.
- 1014 • Cryptographic Usage Mask – This is used to specify the cryptographic operations that the  
 1015 client intends to use the object for (see Section 3.12 ). This allows the server to determine  
 1016 if the policy allows this client to perform these operations with the object. Note that this  
 1017 may be a different value from the one specified in a *Locate* operation that precedes this  
 1018 operation. *Locate*, for example, may specify a Cryptographic Usage Mask requesting a  
 1019 key that can be used for both Encryption and Decryption, but the value in the *Check*  
 1020 operation may specify that the the client is only using the key for Encryption at this time.
- 1021 • Lease Time – This specifies a desired lease time (see Section 3.13 ). The client may use  
 1022 this to determine if the server will allow the client to use the object with the specified lease  
 1023 or longer. Including this attribute in the *Check* operation does not actually cause the  
 1024 server to grant a lease, but only indicates that the requested lease time value will be  
 1025 granted if requested by a subsequent, batched, Obtain Lease operation.

1026 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1027 a Recover operation before they can be specified

<b>Request Payload</b>		
<b>Object</b>	<b>Required Field</b>	<b>Description</b>
Unique Identifier	No	Determines the object being requested. If omitted, 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. May 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. May only be present if Usage Limits Byte Count is not present
Cryptographic Usage Mask	No	Specifies the Cryptographic Usage that the client will use the object for
Lease Time	No	Specifies a Lease Time value that the Client is asking the server to validate against server policy

1028

<b>Response Payload</b>		
<b>Object</b>	<b>Required Field</b>	<b>Description</b>
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 was larger than the value that the server policy would allow. May 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 was larger than the value that the server policy would allow. May 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 was rejected by the server for policy violation
Lease Time	No	Returned by the Server if the Lease Time value in the Request Payload was larger than a valid Lease Time that the server would grant

1029

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

1030

## 4.10 Get

1031 This operation requests that the server returns a Managed Object, which is specified in the  
1032 request by its Unique Identifier. The Unique Identifier field in the request may be omitted if the *Get*  
1033 operation is in a batched set of operations and follows an operation that sets the ID Placeholder  
1034 variable.

1035 Only a single object is returned. Only on-line objects can be specified. Archived objects must first  
1036 be moved back on-line through a Recover operation before they can be specified. The response  
1037 contains the Unique Identifier of the object along with the object itself, which may be optionally  
1038 wrapped using a wrapping key specified in the request.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object being requested. If omitted, the ID Placeholder is substituted by the server
Key Wrapping Specification	No	Specifies keys and other information for wrapping the returned object. This field may not be specified if the returned object is a Template or Policy Template

1039

Response Payload		
Object	Required Field	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, Policy Template, Secret Data, or Opaque Object	Yes	The cryptographic object being returned

1040

## 4.11 Get Attributes

1041 Return one or more attributes of a Managed Object. The object is specified by its Unique  
1042 Identifier. The desired attributes are specified by name in the request. If a specified attribute has  
1043 multiple instances, all instances are returned. If a specified attribute does not exist (i.e. has no  
1044 value) it must not be present in the returned response. If no requested attributes exist, the  
1045 response should consist only of the Unique Identifier. Note that the response payload is empty if  
1046 there are no attribute values to return.

1047 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
1048 a Recover operation before they can be specified.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object whose attributes are being requested. If omitted, the ID Placeholder is substituted by the server
Attribute Name	Yes, May be repeated	Specifies a desired attribute of the object

1049

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	No, May be repeated	The requested attribute for the object

1050

## 4.12 Get Attribute List

1051

Returns a list of the attribute names associated with a specified object. The object is specified by its Unique Identifier.

1052

1053

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

1054

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object whose attribute names are being requested. If omitted, the ID Placeholder is substituted by the server

1055

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute Name	Yes, May be repeated	The requested attribute names for the object

1056

## 4.13 Add Attribute

1057

This request adds a new attribute (with possibly multiple values) and sets its value. The request contains the Unique Identifier of the Managed Object which the attribute pertains to, and the attribute, with its name and new value. For non multi-valued attributes, this is how they are created. For multi-valued attributes, this is how the first and subsequent values are created.

1058

1059

1060

Existing attribute values must be changed by the Modify Attribute operation. Read-Only attributes

1061

1062 may not be added using this operation. No Attribute Index may be specified in the request. The  
 1063 response will return a new Attribute Index if the attribute being added is allowed to have multiple  
 1064 instances. Multiple Add Attribute requests may be included in a single batched request to add  
 1065 multiple attributes.

1066 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1067 a Recover operation before they can be specified.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	The Unique Identifier of the object. If omitted, the ID Placeholder is substituted by the server
Attribute	Yes	Specifies the attribute of the object to be added

1068

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	Yes	The added attribute

#### 1069 4.14 Modify Attribute

1070 This request modifies the value of an existing attribute. The request contains the Unique Identifier  
 1071 of the Managed Object which the attribute pertains to, and the attribute, with its name, optional  
 1072 index, and new value. Only existing attributes may be changed via this operation. New attributes  
 1073 must be added by the Add Attribute operation. Read-Only attributes may not be changed using  
 1074 this operation. If an attribute index is specified, only the specified instance is modified. If the  
 1075 attribute has multiple instances and no index is specified in the request, then the index is  
 1076 assumed to be 0. If the attribute does not support multiple instances, the attribute index must not  
 1077 be specified. Using a non-existing attribute index in a modify operation will result in an error.

1078 The Attribute returned in the response may have a value different from the one sent in the  
 1079 request, if the server policy so dictates. The value returned is the value set by the server.

1080 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1081 a Recover operation before they can be specified.

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

1082

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	Yes	The modified attribute

1083

1084

## 4.15 Delete Attribute

1085

1086

1087

1088

1089

1090

1091

1092

This request deletes an attribute. The request contains the Unique Identifier of the Managed Object which the attribute pertains to, and the name, and optionally the Attribute Index of the attribute. Required attributes and Read-Only attributes may not be deleted by this operation. If no Attribute Index is specified, and the Attribute whose name is specified has multiple instances, the operation is rejected. Note that only a single attribute can be deleted at a time. Multiple delete operations (possible batched) are necessary to delete several attributes. Deleting non-existing attributes will result in an error. Using a non-existing attribute index in a delete operation will also result in an error.

1093

1094

Only on-line objects can be specified. Archived objects must first be moved back on-line through a Recover operation before they can be specified.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object whose attributes are being updated. If omitted, the ID Placeholder is substituted by the server
Attribute Name	Yes	Specifies the name of the attribute of the object to be deleted
Attribute Index	No	Specifies the Index of the Attribute

1095

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	Yes	The deleted attribute

1096

## 4.16 Obtain Lease

1097

1098

1099

1100

1101

1102

1103

1104

This request is used to request a new *Lease Time* for a specified cryptographic object. The Lease Time is an interval value that determines when the client's internal cache of information about the object expires and must be renewed. If the returned value of the lease time is zero, then the server is indicating that no lease interval is effective and the client may use the object without any lease time limit. If a client's lease expires, the client must not use the associated cryptographic object until a new lease is obtained. If the server determines that a new lease should not be issued for the specified cryptographic object, then the server should respond to the Obtain Lease request with a Result Status of Failure, and a Result Reason of General Failure.

1105

1106

The response payload for the operation also contains the current value of the Last Changed Date attribute for the object. This may be used by the client to determine if any of the attributes cached

1107 by the client need to be refreshed, by comparing this time to the time when the attributes were  
1108 previously obtained.

1109 The Unique Identifier field in the request may be omitted if the operation is in a batched set of  
1110 operations and follows an operation that sets the ID Placeholder variable.

1111 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
1112 a Recover operation before they can be specified.

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

1113

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object
Lease Time	Yes	An interval (in seconds) determining the amount of time that the object can 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.

## 1114 4.17 Get Usage Allocation

1115 This request is used to obtain an allocation from the current Usage Limits values, to allow the  
1116 client to use the Managed Cryptographic Object for protection purposes. It only applies to  
1117 Managed Cryptographic Objects that can be used for protection purposes (symmetric keys,  
1118 private keys and public keys) and is only valid if the Managed Cryptographic Object has a Usage  
1119 Limits attribute. Usage for process purposes (decryption, verification, etc.) is not limited and  
1120 cannot be allocated. A Managed Cryptographic Object that has a Usage Limits attribute may not  
1121 be used by a client for protection purposes unless an allocation has been obtained using this  
1122 operation. The operation may only be issued during the time that protection is enabled for these  
1123 objects, i.e. after the Activation Date and before the Protect Stop Date. If the operation is issued  
1124 for an object that has no Usage Limits attribute, or is not an object that can be used for protection  
1125 purposes, the server must return a response with a Result Reason of Operation Not Supported.

1126 The fields in the request specify the number of bytes, or number of objects that the client needs to  
1127 protect. Exactly one of the two count fields must be specified in the request. The corresponding  
1128 field, containing the number of bytes, or number of objects that may be protected, is returned in  
1129 the response. If the requested amount is not available, the server may return a smaller amount, or  
1130 may return 0, indicating that the Managed Object may not be used for protection purposes at this  
1131 time. The server must assume that the entire allocated amount has been consumed. Server  
1132 policy may allow the value returned in the response to be different from the value requested.  
1133 Once the entire allocated amount has been consumed, the client may not continue to use the  
1134 Managed Cryptographic Object for protection purposes until a new allocation is obtained.

1135 The Unique Identifier field in the request may be omitted if the operation is in a batched set of



1136 operations and follows an operation that sets the ID Placeholder variable.  
 1137 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1138 a Recover operation before they can be specified.

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

1139

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object
Usage Limits Byte Count	No	The number of bytes that may be protected. May only be present if Usage Limits Object Count is not present
Usage Limits Object Count	No	The number of objects that may be protected. May only be present if Usage Limits Byte Count is not present

1140 The encodings of the Usage limits Byte and Object Counts is as shown in Section 3.14 .

## 1141 **4.18 Activate**

1142 This request is used to activate a Managed Cryptographic Object. The request may not specify a  
 1143 Template or Policy Template object. The request contains the unique identifier of the Managed  
 1144 Cryptographic Object . The operation can be performed only on an object in the Pre-Active state  
 1145 and has the effect of changing its state to Active and its Activation Date will be set to the current  
 1146 date and time.

1147 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1148 a Recover operation before they can be specified.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object being activated. If omitted, the ID Placeholder is substituted by the server

1149

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object

1150

## 4.19 Revoke

1151 This request is used to revoke a Managed Cryptographic Object. The request may not specify a  
 1152 Template or Policy Template object. The request contains the unique identifier of the Managed  
 1153 Cryptographic Object and a reason for the revocation, e.g. "compromised", "no longer used", etc.  
 1154 Special authentication and authorization is required to issue this request (see Usage Guide). Only  
 1155 the object creator or an authorized security officer should be allowed to issue this request. The  
 1156 operation will have one of two effects. If the revocation reason is "compromised", then the object  
 1157 will be placed into the "compromised" state, and the Compromise Date attribute will be set to the  
 1158 current date and time. Otherwise, the object will be placed into the "deactivated" state, and the  
 1159 Deactivation Date attribute will be set to the current date and time.

1160 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1161 a Recover operation before they can be specified.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object being revoked. If omitted, the ID Placeholder is substituted by the server
Revocation Reason	Yes	Specifies the reason for revocation
Compromise Occurrence Date	No	Only specified, and required, if the Revocation Reason is 'compromised'

1162

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object

1163

## 4.20 Destroy

1164 This request is used to indicate to the server that the key material for the specified Managed  
 1165 Cryptographic Object should be destroyed. The meta-data for the key material may be retained  
 1166 by the server. This is used for example, to ensure that copies of an expired or revoked private  
 1167 signing key are no longer available. Special authentication and authorization is required to issue  
 1168 this request (see Usage Guide). Only the object creator or an authorized security officer should  
 1169 be allowed to issue this request. If the Unique Identifier specifies a Template or Policy Template  
 1170 object, then the object itself, including all meta-data may be destroyed.

1171 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1172 a Recover operation before they can be specified.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object being destroyed.

		If omitted, the ID Placeholder is substituted by the server
--	--	---

1173

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object

1174

## 4.21 Archive

1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182

This request is used to specify that a Managed Object is now permitted to be placed in archival storage. The actual time when the object is placed in archival storage and the location of the archive or level of archive hierarchy is determined by the policies within the key management system, and is not specified by the client. The request contains the unique identifier of the object. Special authentication and authorization is required to issue this request (see Usage Guide). Only the object creator or an authorized security officer should be allowed to issue this request. This request may be considered only a “hint” to the key management system, which may or may not choose to act upon this request.

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object being archived. If omitted, the ID Placeholder is substituted by the server

1183

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object

1184

## 4.22 Recover

1185  
1186  
1187  
1188  
1189

This request is used to obtain access to a Managed Object that has been placed in archival storage. Due to the fact that the object is located in archival storage, this request may require asynchronous polling to obtain the response. Once the response is received, the object is now on-line, and may be obtained via a normal Get operation, for instance. Special authentication and authorization is required to issue this request (see Usage Guide).

Request Payload		
Object	Required Field	Description
Unique Identifier	No	Determines the object being recovered. If omitted, the ID Placeholder is substituted by the server

1190

Response Payload		
Object	Required Field	Description
Unique Identifier	Yes	The Unique Identifier of the object

1191 **4.23 Validate**

1192 This requests that the server validate a certificate chain, and return information on its validity.  
 1193 Only a single certificate chain may be included in each request. Support for this operation at the  
 1194 server is optional.

1195 The request may contain a list of certificate objects, and/or a list of Unique Identifiers which  
 1196 identify Managed Certificate objects. The two lists must together comprise a certificate chain to be  
 1197 validated. The request may also optionally contain a date for which the certificate chain must be  
 1198 valid.

1199 The validation method or policy by which validation will be conducted is a decision of the server  
 1200 and is outside of the scope of this protocol. Likewise, the order in which the supplied certificate  
 1201 chain is validated and the specification of trust anchors used to terminate validation are also  
 1202 controlled by the server.

1203 Only on-line objects can be specified. Archived objects must first be moved back on-line through  
 1204 a Recover operation before they can be specified.

Request Payload		
Object	Required Field	Description
Certificate	No, May be repeated	One or more Certificates
Unique Identifier	No, May be repeated	One or more Unique Identifiers of Certificate Objects
Validity Date	No	A Date-Time object indicating when the certificate chain must be valid

1205

Response Payload		
Object	Required Field	Description
Validity Indicator	Yes	An Enumeration object indicating whether the certificate chain is valid, invalid, or unknown

1206 **4.24 Query**

1207 This request is used by the client to interrogate the server to determine its capabilities and/or  
 1208 protocol mechanisms. It is recommended that the *Query* operation, used to interrogate server  
 1209 features and functions, be invocable by unauthenticated clients. The *Query Function* field in the  
 1210 request may contain one of the following items:

- 1211 • Query Operations
- 1212 • Query Objects
- 1213 • Query Server Information

1214 One, two, or all three of the above functions may be specified.

1215 The *Operation* fields in the response contain Operation enumerated values, which should list the  
 1216 optionally supported operations that the server supports. These fields should only be returned in  
 1217 the response if the request contains a Query Operations value in the Query Function field. The  
 1218 optional operations are:

- 1219 • Validate
- 1220 • Certify
- 1221 • Re-Certify
- 1222 • Notify
- 1223 • Put

1224 The *Object Type* fields in the response contain Object Type enumerated values, which should list  
 1225 the object types that the server supports. These fields should only be returned in the response if  
 1226 the request contains a *Query Objects* value in the Query Function field. The object types (any of  
 1227 which are optional) are:

- 1228 • Certificate
- 1229 • Symmetric Key
- 1230 • Public Key
- 1231 • Private Key
- 1232 • Split Key
- 1233 • Template
- 1234 • Policy Template
- 1235 • Secret Data
- 1236 • Opaque Object

1237 The *Server Information* field in the response is a structure containing vendor specific fields and/or  
 1238 substructures. This field should only be returned in the response if the request contains a *Query*  
 1239 *Server Information* value in the Query Function field.

1240 Note that the response payload is empty if there are no values to return.

Request Payload		
Object	Required Field	Description
Query Function	Yes, May be Repeated	Determines the information being queried

1241

Response Payload		
Object	Required Field	Description
Operation	No, May be repeated	Specifies an Operation that is supported by the server. Only optional operations should be listed
Object Type	No, May be repeated	Specifies a Managed Object Type that is supported by the server
Vendor Identification	No	Must be returned if Query Server Information is requested. The Vendor Identification must be a text string that uniquely identifies the vendor
Server Information	No	Contains vendor-specific information that may be of interest to the client

1242

## 4.25 Cancel

1243

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

1244

1245

1246

- *Canceled* – The cancel operation succeeded in canceling the pending operation.

1247

- *Unable To Cancel* – The cancel operation is unable to cancel the pending operation.

1248

- *Completed* – The pending operation completed successfully before the cancellation operation was able to cancel it.

1249

1250

- *Failed* – The pending operation completed with a failure before the cancellation operation was able to cancel it.

1251

1252

- *Unavailable* – The specified correlation value did not match any recently pending or completed asynchronous operations.

1253

1254

The response to this operation cannot be asynchronous.

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

1255

Response Payload		
Object	Required Field	Description
Asynchronous Correlation Value	Yes	Specifies the request
Cancellation Result	Yes	Enumeration indicating result of cancellation

1256

## 4.26 Poll

1257

This request is used to poll the server in order to obtain the status of an outstanding asynchronous operation. The correlation value (see Section 6.8 ) of the original operation must be specified in the request. The response to this operation cannot be asynchronous.

1258

1259

Request Payload		
Object	Required Field	Description
Asynchronous Correlation Value.	Yes	Specifies the request being polled

1260 The server must reply with one of two responses:

- 1261 • A response containing no payload and a Result Status of Pending, if the operation
- 1262 has not completed
- 1263 • A response containing the appropriate payload for the operation, if the operation
- 1264 has completed. This response must be identical to the response that would have
- 1265 been sent if the operation had completed synchronously.

## 1266 5 Server-to-Client Operations

1267 Server-to-client operations are used by servers to send information or Managed Cryptographic Objects to  
 1268 clients outside of the normal client-server request-response mechanism. These operations are used to  
 1269 “push” Managed Cryptographic Objects directly to clients without a specific request from the client.

### 1270 5.1 Notify

1271 This operation is used to notify a client of events. This operation is only ever sent by a server to a  
 1272 client outside the normal client request/response protocol, using information known to the server  
 1273 via unspecified configuration or administrative mechanisms. It contains the Unique Identifier of the  
 1274 object to which the notification applies, and a list of the attributes whose changed values have  
 1275 triggered the notification. The message is sent as a normal Request message, except that the  
 1276 Maximum Response Size, Asynchronous Indicator, Batch Error Continuation Option, and Batch  
 1277 Order Option fields are not allowed. The client must send a response in the form of a Response  
 1278 Message containing no payload, unless both the client and server have prior knowledge (obtained  
 1279 via out-of-band mechanisms) that the client cannot respond. Server and Client support for this  
 1280 message is optional.

Message Payload		
Object	Required Field	Description p
Unique Identifier	Yes	The Unique Identifier of the object
Attribute	Yes, May be repeated	The attributes which have changed. This includes at least the Last Changed Date attribute

### 1281 5.2 Put

1282 This operation is used to “push” Managed Cryptographic Objects to clients. This operation is only  
 1283 ever sent by a server to a client outside the normal client request/response protocol, using  
 1284 information known to the server via unspecified configuration or administrative mechanisms. It  
 1285 contains the Unique Identifier of the object which is being sent, and the object itself. The message  
 1286 is sent as a normal Request message, except that the Maximum Response Size, Asynchronous  
 1287 Indicator, Batch Error Continuation Option, and Batch Order Option fields are not allowed. The  
 1288 client must send a response in the form of a Response Message containing no payload, unless  
 1289 both the client and server have prior knowledge (obtained via out-of-band mechanisms) that the  
 1290 client cannot respond. Server and client support for this message is optional.

1291 The *Put Function* field indicates whether the object being “pushed” is a new object, or a  
 1292 replacement for an object already known to the client. For example, when pushing a certificate to

1293 replace one that is about to expire, the Put Function field would be set to indicate replacement,  
 1294 and the Unique Identifier of the expiring certificate would be placed in the *Replaced Unique*  
 1295 *Identifier* field. The Put Function may contain one of the following values:

- 1296 • *New* – which indicate that the object is not a replacement for another object
- 1297 • *Replace* – which indicates that the object is a replacement for another object, and that the  
 1298 Replaced Unique Identifier field is present, and contains the identification of the replaced  
 1299 object.

1300 The Attribute field contains one or more attributes that the server wishes to be pushed along with  
 1301 the object. In particular, the server may include policy attributes with the object to specify how the  
 1302 object is to be used by the client. The server may include a Lease Time attribute which grants a  
 1303 lease to the client.

1304 If the Managed Object is a wrapped key, the key wrapping specification must be exchanged prior  
 1305 to the transfer via out-of-band mechanisms.

Message Payload		
Object	Required Field	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 replaced object. Must be present if the <i>Put Function</i> is <i>Replace</i>
Certificate, Symmetric Key, Private Key, Public Key, Split Key, Template, Policy 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 push with the object

## 1306 6 Message Contents

1307 Messages in the protocol consist of a message header and one or more batch items which contain  
 1308 optional message payloads, and optional message extensions. The message headers contain fields  
 1309 whose presence is determined by the protocol features used, e.g. asynchronous responses. The field  
 1310 contents are also determined by whether the message is a request or a response. The message payload  
 1311 is determined by the specific operation being requested or replied to.

1312 The message headers are structures which contain some of the following objects.

### 1313 6.1 Protocol Version

1314 This field contains the version number of the protocol, ensuring that the protocol is fully  
 1315 understood by both communicating parties. The version number is specified in two parts, major  
 1316 and minor. Servers and clients must support backward compatibility with versions of the protocol  
 1317 with the same major version but different minor versions. Support for backward compatibility with  
 1318 different major versions is optional.



Object	Encoding	Required Field
Protocol Version	Structure	Yes
Protocol Version Major	Integer	Yes
Protocol Version Minor	Integer	Yes

1319 **6.2 Operation**

1320 This field indicates the operation being requested or the operation for which the response is being  
1321 returned. The operations are defined in Sections 4 and 5 .

Object	Encoding	Required Field
Operation	Enumeration	Yes

1322 **6.3 Maximum Response Size**

1323 This field is optionally contained in a request message, and is used to indicate the maximum size  
1324 of a response that the requester can handle. It need only be sent in requests that may return  
1325 large replies.

Object	Encoding	Required Field
Maximum Response Size	Integer	No

1326 **6.4 Unique Message ID**

1327 This field is optionally contained in a request, and is used as for correlation between requests and  
1328 responses. If a request has a *Unique Message ID*, then responses to that request must have the  
1329 same Unique Message ID.

Object	Encoding	Required Field
Unique Message ID	Octet String	No

1330 **6.5 Time Stamp**

1331 This field is optionally contained in a request and required in a response, and is used for time  
1332 stamping and may be used to enforce reasonable time usage at a client, e.g. a server may  
1333 choose to reject a request if a client's time stamp contains a value that is too far off the known  
1334 correct time. It may also be used by a client, which has no real-time clock but only a countdown  
1335 timer, to obtain useful "seconds from now" values from all of the Date attributes, by performing a  
1336 subtraction.

Object	Encoding	Required Field
Time Stamp	Date-Time	No

1337 **6.6 Authentication**

1338 This is used to authenticate the requester. It is an optional information item, depending on the  
1339 type of request being issued and on server policies. Servers may require authentication on no  
1340 requests, a subset of the operations, or all requests, depending on policy. It is recommended that  
1341 the Query operation, used to interrogate server features and functions, not require authentication.

1342 The authentication mechanisms are described and discussed in Section 8 .

Object	Encoding	Required Field
Authentication	Structure	No
Credential	Structure	Yes

1343 The Credential structure is defined in Section 2.1.2 .

## 1344 6.7 Asynchronous Indicator

1345 This boolean flag indicates whether the client can accept an asynchronous response. It must  
 1346 have the boolean value True if the client can handle asynchronous responses, and the value  
 1347 False otherwise. If not present in a request, False is assumed. If a client indicates that it can't  
 1348 handle asynchronous responses (flag is set to False) and the server is not capable to process the  
 1349 request synchronously, the server must reject the request with a failure status.

Object	Encoding	Required Field
Asynchronous Indicator	Boolean	No

## 1350 6.8 Asynchronous Correlation Value

1351 This is returned in the immediate response to an operation that will require asynchronous polling.  
 1352 It is a server generated correlation value that must be specified in any subsequent Poll, or Cancel  
 1353 operations that pertain to the original operation.

Object	Encoding	Required Field
Asynchronous Correlation Value	Octet String	No

## 1354 6.9 Result Status

1355 This indicates the success or failure of the request. The following values may be set in this field:

- 1356 • *Success* – The requested operation completed successfully.
- 1357 • *Pending* – The requested operation is in progress and the actual result must be obtained  
 1358 via asynchronous polling. The asynchronous correlation value must be used for the  
 1359 subsequent polling of the result status.
- 1360 • *Undone* – The requested operation was performed but had to be undone (due to a  
 1361 failure in a batch for which the Error Continuation Option was set to Undo)
- 1362 • *Failure* – The requested operation failed.

Object	Encoding	Required Field
Result Status	Enumeration	Yes

## 1363 6.10 Result Reason

1364 This field indicates a reason for failure or a modifier for a partially successful operation and must  
 1365 be present in responses that return a Result Status of Failure. It is optional in any response that  
 1366 returns a Result Status of Success. The following defined values may be set in this field:

- 1367 • *Item Not Found* – A requested object was not found or did not exist.
- 1368 • *Response too large* – The response to a request would have exceeded the *Maximum*  
 1369 *Response Size* in the request.
- 1370 • *Authentication not successful* – The authentication in the request did not pass validation,  
 1371 or there was no authentication in the request when there should have been.

- 1372 • *Invalid Message* – The request message was not understood by the server.
- 1373 • *Operation Not Supported* – The operation requested by the request message was not supported by the server.
- 1374
- 1375 • *Missing Data* – The operation required additional optional information in the request, which was not present.
- 1376
- 1377 • *Invalid Field* – Some data item in the request had an invalid value.
- 1378 • *Feature not supported* – An optional feature specified in the request was not supported.
- 1379 • *Operation canceled by requester* – The operation was asynchronous and the operation was canceled by the Cancel operation before it completed successfully.
- 1380
- 1381 • *Cryptographic failure* – The operation failed due to a cryptographic error.
- 1382 • *Illegal Operation* – The client requested an operation that could not be performed with the specified parameters.
- 1383
- 1384 • *Permission Denied* – The client did not have permission to perform the requested operation.
- 1385
- 1386 • *Object archived* – The object should be first recovered from the archive.
- 1387 • *General Failure* – The request failed for a reason other than the defined reasons above.

Object	Encoding	Required Field
Result Reason	Enumeration	Yes

## 1388 6.11 Result Message

1389 This field may optionally be returned in a response. It contains a more descriptive error message, which may be used by the client to display to an end user or for logging/auditing purposes.

1390

Object	Encoding	Required Field
Result Message	Text String	No

## 1391 6.12 Batch Order Option

1392 A Boolean value used in requests where the Batch Count is greater than 1. If true then batched operations must be executed in the order in which they appear within the request. If false, the server may choose to execute the batched operations in any order. If not specified, false is assumed (i.e. no implied ordering). Server support for this feature is optional, but if the server does not support the feature, and a request is received with the flag true, the entire request must be rejected.

1393

1394

1395

1396

1397

Object	Encoding	Required Field
Batch Order Option	Boolean	No

## 1398 6.13 Batch Error Continuation Option

1399 Batched operation partial failure continuation option. This option should only be present if the Batch Count is greater than 1. This option may have one of three values:

1400

- 1401 • *Undo* – If any operation in the request fails, the server must undo all the previous operations.
- 1402 • *Stop* – If an operation fails, the server must not continue processing later operations in the request. Completed operations will be left intact.
- 1403
- 1404 • *Continue* – Return an error for the failed operation and continue processing later operations in the request.
- 1405

1406 If not specified, Stop is assumed.

1407 Server support for this feature is optional, but if the server does not support the feature, and a  
1408 request is received containing the *Batch Error Continuation* option, the entire request must be  
1409 rejected.

Object	Encoding	Required Field
Batch Error Continuation Option	Enumeration	No

## 1410 6.14 Batch Count

1411 This field is required. It contains the number of Batch Items in a message. If only a single  
1412 operation is being requested, the batch count must be set to 1. The Message Payload, which  
1413 follows the Message Header, will contain one or more batch items.

Object	Encoding	Required Field
Batch Count	Integer	Yes

## 1414 6.15 Batch Item

1415 This field is required. It consists of a structure that holds the individual requests or responses in a  
1416 batch. The contents of the batch items is described in Sections 7.2 and 7.3 .

Object	Encoding	Required Field
Batch Item	Structure	No

## 1417 6.16 Message Extension

1418 The *Message Extension* is an optional structure which may be appended to any Batch Item. It is  
1419 used to extend protocol messages for the purpose of adding vendor specified extensions. The  
1420 Message Extension is a structure containing a Vendor Identification, Criticality Indicator, and  
1421 vendor-specific extensions. The *Vendor Identification* must be a text string that uniquely identifies  
1422 the vendor, allowing a client to determine if the extension can be parsed and understood. If a  
1423 client or server receives a protocol message containing a message extension that it does not  
1424 understand, its actions depend on the *Criticality Indicator*. If the indicator is True (Critical), and the  
1425 receiver does not understand the extension, the receiver must reject the entire message. If the  
1426 indicator is False (Non-Critical), and the receiver does not understand the extension, the receiver  
1427 may process the rest of the message as if the extension were not present.

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

## 1428 7 Message Format

1429 Messages contain the following objects and fields. All fields must appear in the order specified.

1430

## 7.1 Message Structure

Object	Encoding	Required Field
Request Message	Structure	Yes
Request Header	Structure	Yes
Batch Item	Structure	Yes, May be repeated

1431

Object	Encoding	Required Field
Response Message	Structure	Yes
Response Header	Structure	Yes
Batch Item	Structure	Yes, May be repeated

1432

## 7.2 Synchronous Operations

Synchronous Request Header		
Object or Field	Required in Message	Comment
Request Header	Yes	Structure
Protocol Version	Yes	
Maximum Response Size	No	
Authentication	No	
Batch Error Continuation Option	No	If omitted, Stop is assumed
Batch Order Option	No	If omitted, False is assumed
Time Stamp	No	
Batch Count	Yes	

1433

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

1434

Synchronous Response Header		
Object or Field	Required in Message	Comment
Response Header	Yes	Structure
Protocol Version	Yes	
Time Stamp	Yes	
Batch Count	Yes	

1435

Synchronous Response Batch Item		
Object or Field	Required in Message	Comment
Batch Item	Yes	Structure
Operation	Yes, if not a failure	
Unique Message ID	No	Required 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	

1436

### 7.3 Asynchronous Operations

1437

1438

1439

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

Asynchronous Request Header		
Object or Field	Required in Message	Comment
Request Header	Yes	Structure
Protocol Version	Yes	
Maximum Response Size	No	
Asynchronous Indicator	Yes	Must be set to True
Authentication	No	
Batch Error Continuation Option	No	If omitted, Stop is assumed
Batch Order Option	No	If omitted, False is assumed
Time Stamp	No	
Batch Count	Yes	

1440

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

1441

Asynchronous Response Header		
Object or Field	Required in Message	Comment
Response Header	Yes	Structure
Protocol Version	Yes	
Time Stamp	Yes	
Batch Count	Yes	

1442

Asynchronous Response Batch Item		
Object or Field	Required in Message	Comment
Batch Item	Yes	Structure
Operation	Yes, if not a failure	
Unique Message ID	No	Required 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	

## 1443 8 Authentication

1444 The mechanisms used to authenticate the client to the server and the server to the client are not part of  
 1445 the message definitions, and are external to the protocol. The *Authentication* field contained in Request  
 1446 Headers is used to identify the client and to provide linkage between this identification and the external  
 1447 authentication mechanism.

1448 The Usage Guide describes authentication profiles appropriate to this protocol as well as the relationship  
 1449 of those mechanisms to the credentials optionally included in the Authentication field. The authentication  
 1450 profiles described are:

- 1451
- 1452
- 1453
- 1454
- 1455
- 1456
- SSL/TLS authentication. If the transport protocol uses a normal TCP stream, then that stream should use an SSL/TLS encryption layer and the client and server authentication features must be enabled. The Credential object contained in the Authentication field in all request messages will contain the client's certificate. The server should use this certificate to identify the client for policy enforcement purposes, and should verify that this certificate matches the one used for SSL/TLS authentication.
- 1457
- HTTPS authentication. If the transport protocol is HTTP over TCP, then the HTTPS protocol should be used, and the client and server authentication features enabled. The contents and use of the *Credential* object are the same as in the normal TCP example above.
- 1458
- 1459
- 1460 All server implementations should, at least, support the SSL/TLS and HTTPS profiles described in the
- 1461 Usage Guide.
- 1462 Other mechanisms, such as Kerberos, are potentially usable, with the identity established in the
- 1463 mechanism, such as the Kerberos token, expressed as the Credential object. Profiles for these
- 1464 mechanisms currently are not described in the Usage Guide.

## 1465 9 Message Encoding

1466 To support different transport protocols and different client capabilities, a number of message-encoding

1467 mechanisms are supported.

### 1468 9.1 TTLV Encoding

1469 In order to minimize the resource impact on potentially low-function clients, one encoding

1470 mechanism to be used for protocol messages is a simplified TTLV (Tag, Type, Length, Value)

1471 scheme.

1472 The scheme is designed to minimize the CPU cycle and memory requirements of clients that

1473 must encode or decode protocol messages, and to provide optimal alignment for both 32-bit and

1474 64-bit processors. Minimizing bandwidth over the transport mechanism is considered to be of

1475 lesser importance.

#### 1476 9.1.1 TTLV Encoding Fields

1477 Every Data object encoded by the TTLV scheme consists of 4 items, in order:

##### 1478 9.1.1.1 Item Tag

1479 An Item Tag is a 3-byte binary unsigned integer, transmitted big endian, which contains a

1480 number that designates the specific Protocol Field or Object that the TTLV object

1481 represents. To ease debugging, and to ensure that malformed messages are detected

1482 more easily, all tags must contain either the value 42 in hex or the value 54 in hex as the

1483 high order (first) byte. Tags defined by this specification contain hex 42 in the first byte.

1484 Extensions, which are permitted, but not defined in this specification, contain the value 54

1485 hex in the first byte. A list of defined Item Tags is in Section 9.1.3.1 .

##### 1486 9.1.1.2 Item Type

1487 An Item Type is a byte containing a coded value that indicates the data type of the data

1488 object. The allowed values are:



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

1489

### 9.1.1.3 Item Length

1490

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

1491

1492

Allowed values are:

1493

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

1494 If the Item Type is Structure, then the Item Length is the total length of all of the sub-  
1495 items contained in the structure, including any padding. If the Item Type is Integer,  
1496 Enumeration, Text String, Octet String, or Interval, then the Item Length is the number of  
1497 bytes excluding the padding bytes. Text Strings and Octet Strings must be padded with  
1498 the minimal number of bytes following the Item Value to obtain a multiple of 8 bytes.  
1499 Integers, Enumerations, and Intervals must be padded with 4 bytes following the Item  
1500 Value.

#### 1501 9.1.1.4 Item Value

1502 The item value is a sequence of bytes containing the value of the data item, depending  
1503 on the type:

- 1504 • Integers are encoded as 4-byte long (32 bit) binary signed numbers in 2's  
1505 complement notation, transmitted big-endian.
- 1506 • Long Integers are encoded as 8-byte long (64 bit) binary signed numbers in 2's  
1507 complement notation, transmitted big-endian.
- 1508 • Big Integers are encoded as a sequence of 8-bit bytes, in 2's complement  
1509 notation, transmitted big-endian. If the length of the sequence is not a multiple of  
1510 8 bytes, then Big Integers shall be padded with the minimal number of leading  
1511 sign-extended bytes to make the length a multiple of 8 bytes. These padding  
1512 bytes are part of the Item Value and must be counted in the Item Length.
- 1513 • Enumerations are encoded as 4-byte long (32 bit) binary unsigned numbers  
1514 transmitted big-endian. Extensions, which are permitted, but not defined in this  
1515 specification, contain the value 8 hex in the first nibble of the first byte.
- 1516 • Booleans are encoded as an 8-byte value that must either contain the hex value  
1517 0000000000000000, indicating the boolean value *False*, or the hex value  
1518 0000000000000001, transmitted big-endian, indicating the boolean value *True*.
- 1519 • Text Strings are sequences of bytes encoding character values according to the  
1520 UTF-8 encoding standard. There must be no null-termination at the end of such  
1521 strings.
- 1522 • Octet Strings are sequences of bytes containing individual unspecified 8 bit  
1523 binary values.
- 1524 • Date-Time values are encoded as 8-byte long (64 bit) binary signed numbers,  
1525 transmitted big-endian. They are POSIX Time values (described in IEEE  
1526 Standard 1003.1) extended to a 64 bit value to eliminate the "Year 2038  
1527 problem". The value is expressed as the number of seconds from a time epoch,  
1528 which is 00:00:00 GMT January 1<sup>st</sup>, 1970. This value has a resolution of 1  
1529 second. All Date-Time values are expressed as UTC values.
- 1530 • Intervals are encoded as 4-byte long (32 bit) binary unsigned numbers,  
1531 transmitted big-endian. They have a resolution of 1 second.
- 1532 • Structure Values are encoded as the concatenated encodings of the elements of  
1533 the structure. All structures defined in this specification must have all of their  
1534 fields encoded in the order in which they appear in their respective structure  
1535 descriptions.

#### 1536 9.1.2 Examples

1537 These examples are assumed to be encoding a Protocol Object whose tag is 420020. The  
1538 examples are shown as a sequence of bytes in hexadecimal notation:

- 1539 • An Integer containing the decimal value 8:  
1540 42 00 20 | 02 | 00 00 00 04 | 00 00 00 08 00 00 00 00

- 1541 • A Long Integer containing the decimal value 1234567890000000000:
- 1542 42 00 20 | 03 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00
- 1543 • A Big Integer containing the decimal value 1234567890000000000000000000000000000000:
- 1544 42 00 20 | 04 | 00 00 00 10 | 00 00 00 00 03 FD 35 EB 6B C2 DF 46
- 1545 18 08 00 00
- 1546 • An Enumeration with value 255:
- 1547 42 00 20 | 05 | 00 00 00 04 | 00 00 00 FF 00 00 00 00
- 1548 • A Boolean with the value *True*:
- 1549 42 00 20 | 06 | 00 00 00 08 | 00 00 00 00 00 00 00 01
- 1550 • A Text String:
- 1551 42 00 20 | 07 | 00 00 00 0B | 48 65 6C 6C 6F 20 57 6F 72 6C 64 00
- 1552 00 00 00 00
- 1553 • An Octet String:
- 1554 42 00 20 | 08 | 00 00 00 03 | 01 02 03 00 00 00 00 00
- 1555 • A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:
- 1556 42 00 20 | 09 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8
- 1557 • An Interval, containing the value for 10 days:
- 1558 42 00 20 | 0A | 00 00 00 04 | 00 0D 2F 00 00 00 00 00
- 1559 • A Structure containing an Enumeration, value 254, followed by an Integer, value 255,
- 1560 having tags 420004 and 420005 respectively:
- 1561 42 00 20 | 01 | 00 00 00 20 | 42 00 04 | 05 | 00 00 00 04 | 00 00
- 1562 00 FE 00 00 00 00 | 42 00 05 | 02 | 00 00 00 04 | 00 00 00 FF 00
- 1563 00 00 00

### 9.1.3 Defined Values

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

#### 9.1.3.1 Tags

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

Tag	
Object	Tag Value
(Unused)	000000 - 420000
Activation Date	420001
Application Identifier	420002
Application Name Space	420003
Application Specific Identification	420004
Archive Date	420005
Asynchronous Correlation	420006

<b>Tag</b>	
<b>Object</b>	<b>Tag Value</b>
Value	
Asynchronous Indicator	420007
Attribute	420008
Attribute Index	420009
Attribute Name	42000A
Attribute Value	42000B
Authentication	42000C
Batch Count	42000D
Batch Error Continuation Option	42000E
Batch Item	42000F
Batch Order Option	420010
Block Cipher Mode	420011
Cancellation Result	420012
Certificate	420013
Certificate 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

<b>Tag</b>	
<b>Object</b>	<b>Tag Value</b>
Cryptographic Usage Mask	420028
Custom Attribute	420029
D	42002A
Deactivation Date	42002B
Derivation Data	42002C
Derivation Method	42002D
Derivation Parameters	42002E
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

<b>Tag</b>	
<b>Object</b>	<b>Tag Value</b>
Message Extension	42004C
Modulus	42004D
Name	42004E
Name Type	42004F
Name Value	420050
Object Group	420051
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
Policy Template	42005B
Prime Exponent P	42005C
Prime Exponent Q	42005D
Prime Field Size	42005E
Private Exponent	42005F
Private Key	420060
Private Key Template-Attribute	420061
Private Key Unique Identifier	420062
Process Start Date	420063
Protect Stop Date	420064
Protocol Version	420065
Protocol Version Major	420066
Protocol Version Minor	420067
Public Exponent	420068
Public Key	420069
Public Key Template-Attribute	42006A
Public Key Unique Identifier	42006B
Put Function	42006C
Q	42006D
Q String	42006E
Query Function	42006F

<b>Tag</b>	
<b>Object</b>	<b>Tag Value</b>
Recommended Curve	420070
Replaced Unique Identifier	420071
Request Header	420072
Request Message	420073
Request Payload	420074
Response Header	420075
Response Message	420076
Response Payload	420077
Result Message	420078
Result Reason	420079
Result Status	42007A
Revocation Message	42007B
Revocation Reason	42007C
Revocation Reason Code	42007D
Role Type	42007E
Salt	42007F
Secret Data	420080
Secret Data Type	420081
Serial Number	420082
Server Information	420083
Split Key	420084
Split Key Method	420085
Split Key Parts	420086
Split Key Threshold	420087
State	420088
Storage Status Mask	420089
Symmetric Key	42008A
Template	42008B
Template Name	42008C
Template-Attribute	42008D
Time Stamp	42008E
Unique Identifier	42008F
Unique Message ID	420090
Usage Limits	420091
Usage Limits Byte Count	420092
Usage Limits Object Count	420093

<b>Tag</b>	
<b>Object</b>	<b>Tag Value</b>
Usage Limits Total Bytes	420094
Usage Limits Total Objects	420095
Validity Date	420096
Validity Indicator	420097
Vendor Extension	420098
Vendor Identification	420099
Wrapping Method	42009A
X	42009B
Y	42009C
(Reserved)	42009D - 42FFFF
(Unused)	430000 - 53FFFF
Extensions	540000 - 54FFFF
(Unused)	550000 - FFFFFFFF



1571

### 9.1.3.2 Enumerations

1572

The following tables define the values for enumerated lists.

1573

#### 9.1.3.2.1 Credential Type Enumeration

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

1574

#### 9.1.3.2.2 Key Value Type Enumeration

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

1575

### 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
Vendor specific	00000006
Extensions	8XXXXXXXX

1576

### 9.1.3.2.4 Recommended Curves for ECDSA and ECDH

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

1577

### 9.1.3.2.5 Certificate Type Enumeration

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

1578

**9.1.3.2.6 Split Key Method Enumeration**

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

1579

**9.1.3.2.7 Secret Data Type Enumeration**

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

1580

**9.1.3.2.8 Opaque Data Type Enumeration**

Opaque Data Type	
Name	Value
Extensions	8XXXXXXXX

1581

**9.1.3.2.9 Name Type Enumeration**

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

1582

### 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
Policy Template	00000007
Secret Data	00000008
Opaque Object	00000009
Extensions	8XXXXXXXX

1583

### 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

1584

### 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
AESKeyWrap	0000000B
Extensions	8XXXXXXXX

1585

### 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
Extensions	8XXXXXXXX

1586

### 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	8XXXXXXXX

1587

### 9.1.3.2.15 Role Type Enumeration

Role Type	
Name	Value
ZMK	00000001
ZPK	00000002
MAC	00000003
CVK	00000004
CSC	00000005
PVKIBM	00000006
PVKPVV	00000007
MKCVC	00000008
MKSMI	00000009
MKSMC	0000000A
MKIDN	0000000B
MKAC	0000000C
MKCAP	0000000D
BDK	0000000E
Extensions	8XXXXXXXX

1588

**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

1589

**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

1590

**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

1591 Note: Link Types start at 101 to avoid any confusion with Object Types.

1592 **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

1593 **9.1.3.2.20 Certificate Request Type Enumeration**

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

1594 **9.1.3.2.21 Validity Indicator Enumeration**

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

1595 **9.1.3.2.22 Query Function Enumeration**

Query Function	
Name	Value
Query Operations	00000001
Query Objects	00000002
Query Server Information	00000003
Extensions	8XXXXXXXX



1596

### 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

1597

### 9.1.3.2.24 Put Function Enumeration

Put Function	
Name	Value
New	00000001
Replace	00000002
Extensions	8XXXXXXXX

### 9.1.3.2.25 Operations Enumeration

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

1599

### 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

1600

### 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
General Failure	00000100
Extensions	8XXXXXXXX

1601

### 9.1.3.2.28 Batch Error Continuation Enumeration

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

1602

### 9.1.3.3 Bit Masks

1603

#### 9.1.3.3.1 Cryptographic Usage Mask Values

Cryptographic Usage Mask	
Name	Value
Sign	00000001
Verify	00000002
Encrypt	00000004
Decrypt	00000008
Wrap	00000010
Unwrap	00000020
Export	00000040
MAC	00000080
Derive Key	00000100
Content Commitment (Non Repudiation)	00000200
Key Agreement	00000400
Certificate Sign	00000800
CRL Sign	00001000
MAC Verify	00002000
Extensions	XXXX0000

1604

1605

1606

This list takes into consideration values which may appear in the Key Usage extension in an X.509 certificate. However, the list does not consider the more fine grained usages which may appear in the Extended Key Usage extension.

1607

#### 9.1.3.3.2 Storage Status Mask

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

1608

## 9.2 XML Encoding

1609

An XML Encoding has not yet been defined.

1610

## 10 Transport

1611

1612

1613

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:

1614

1615

- 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.

- 1616 • HTTPS over TCP. This profile describes the implementation of this protocol using HTTPS, with  
1617 client and server authentication features enabled, over a normal TCP stream.

1618 To ensure a base level of interoperability, all server implementations should, at least, support the  
1619 SSL/TLS and HTTPS transport protocols as described in the Usage Guide.

## 1620 11 Error Handling

1621 This section details the specific Result Reasons that should be returned for errors detected. Note that this  
1622 is not an exhaustive list of possible errors for each operation (allowing other Result Reasons to be  
1623 returned if an implementation needs to do so).

### 1624 11.1 General

1625 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 (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 (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 cannot 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

1626

## 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 (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

1627

## 11.3 Create Key Pair

Error Definition	Result Status	Result Reason
Templates that do not exist are given in request	Operation Failed	Item Not Found
Incorrect attribute value(s) specified	Operation Failed	Invalid Field
Error creating cryptographic object (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
Required field(s) missing	Operation Failed	Invalid Message

1628

## 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

1629

## 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 cannot be re-keyed (not a symmetric key or the permissions do not allow it)	Operation Failed	Permission Denied
Offset field cannot 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
Object is archived	Operation Failed	Object archived

1630

## 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 cannot 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
One or more of the objects is archived	Operation Failed	Object archived

1631

## 11.7 Certify

Error Definition	Result Status	Result Reason
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object specified cannot be certified (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
Object is archived	Operation Failed	Object archived

1632

## 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 cannot be certified (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 cannot be specified at the same time as any of the Activation Date or Deactivation Date attributes	Operation Failed	Invalid Message
Object is archived	Operation Failed	Object archived

1633

## 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

1634

## 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



1635

## 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 cannot 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 cannot be used for MAC'ing/signing	Operation Failed	Permission Denied
No cryptographic material associated with object	Operation Failed	Illegal Operation
Cryptographic Parameters associated with object do not exist or do not match with those provided in the Encryption Key Information and/or Signature Key Information	Operation Failed	Item Not Found
Object is archived	Operation Failed	Object archived

1636

## 11.12 Get Attributes

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

1637

## 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

1638

## 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 index	Operation Failed	Invalid Field
Trying to add a Name attribute with the same value that another object already has	Operation Failed	Illegal Operation
Object is archived	Operation Failed	Object archived

1639

## 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 (must 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 something that another object already has	Operation Failed	Illegal Operation
Object is archived	Operation Failed	Object archived

1640

## 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 and therefore no index	Operation Failed	Item Not Found
No attribute with specified name exists	Operation Failed	Item Not Found
Object is archived	Operation Failed	Object archived

1641

**11.17 Obtain Lease**

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

1642

**11.18 Get Usage Allocation**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object has no Usage Limits attribute or object cannot be used for protection purposes	Operation Failed	Illegal Operation
Both Usage Limits Byte Count and Usage Limits Object Count fields 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 can only be given for the other type	Operation Failed	Operation Not Supported
Object is archived	Operation Failed	Object archived

1643

**11.19 Activate**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Unique Identifier specifies template, policy template or other object that cannot be revoked	Operation Failed	Illegal Operation
Object is not in Pre-Active state	Operation Failed	Permission Denied
Object is archived	Operation Failed	Object archived

1644

**11.20 Revoke**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Revocation Reason is not recognized	Operation Failed	Invalid Field
Unique Identifier specifies template, policy template or other object that cannot be revoked	Operation Failed	Illegal Operation
Object is archived	Operation Failed	Object archived

1645

**11.21 Destroy**

<b>Error Definition</b>	<b>Result Status</b>	<b>Result Reason</b>
No object with the specified Unique Identifier exists	Operation Failed	Item Not Found
Object 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

1646

**11.22 Archive**

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

1647

**11.23 Recover**

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

1648

**11.24 Validate**

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

1649

**11.25 Query**

1650

N/A

1651 **11.26 Cancel**

1652 N/A

1653 **11.27 Poll**

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

1654 **11.28 Batch Items**

1655 These errors may occur when a protocol message with one or more batch items is processed by  
1656 the server. If a message with one or more than a single batch item was parsed correctly, the  
1657 response message should include response(s) to the batch item(s) in the request according to  
1658 the table below.

1659

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

1660 **12 Security Considerations**

1661 TBD

1662

## A. Attribute Cross-reference

1663

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

1664

Note that this table is not normative.

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

	Managed Object								
Last Changed Date	x	x	x	x	x			x	x
Custom Attribute	x	x	x	x	x	x	x	x	x

1665

1666

## B. Tag Cross-reference

1667

Note that this table is not normative.

Object	Defined	Type	Notes
Activation Date	3.17	Date-Time	
Application Identifier	3.28	Text String	
Application Name Space	3.28	Text String	
Application Specific Identification	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	3.22	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	
MAC/Signature	2.1.5	Octet String	

Object	Defined	Type	Notes
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	2.2.9 , 9.1.3.2.8	Enumeration	
Opaque Data Value	2.2.9	Octet String	
Opaque Object	2.2.9	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	
Policy Template	2.2.7	Structure	
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	
Query Function	4.24 , 9.1.3.2.22	Enumeration	

Object	Defined	Type	Notes
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	2.2.8	Structure	
Secret Data Type	2.2.8 , 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 Name	4.3	Text String	
Template-Attribute	2.1.8	Structure	
Time Stamp	6.5	Date-Time	
Transparent*	2.1.7	Structure	
Unique Identifier	3.1	Text String	
Unique Message 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	

<b>Object</b>	<b>Defined</b>	<b>Type</b>	<b>Notes</b>
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	

1668 **C. Operation and Object Cross-reference**

1669 The following table indicates the types of Managed Object(s) that each Operation can take as input or  
 1670 provide as output. Note that this table is not normative.

1671

Operation	Managed Objects							
	Certificate	Symmetric Key	Public Key	Private Key	Split Key	Template	Policy Template	Secret Data
Create	N/A	Y	N/A	N/A	N/A	Y	Y	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	N/A	N/A	N/A
Derive Key	N/A	Y	N/A	N/A	N/A	N/A	N/A	Y
Certify	Y	N/A	Y	N/A	N/A	N/A	N/A	N/A
Re-certify	Y	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Locate	Y	Y	Y	Y	Y	Y	Y	Y
Check	Y	Y	Y	Y	Y	N/A	N/A	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	N/A	Y
Get Usage Allocation	N/A	Y	Y	Y	N/A	N/A	N/A	N/A
Activate	Y	Y	Y	Y	Y	N/A	N/A	Y
Revoke	Y	Y	N/A	Y	Y	N/A	N/A	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

## 1672 **D. Acronyms**

1673 The following abbreviations and acronyms are used in this document:

- 1674 3DES - Three key Data Encryption Standard
- 1675 AES - Advanced Encryption Standard specified in FIPS 197
- 1676 ASN.1 - Abstract Syntax Notation One
- 1677 CA - Certification Authority
- 1678 CBC - Cipher Block Chaining
- 1679 CPU - Central Processing Unit
- 1680 CRL - Certificate Revocation List
- 1681 CRT - Chinese Remainder Theorem
- 1682 DER - Distinguished Encoding Rules
- 1683 DES - Data Encryption Standard
- 1684 DH - Diffie-Hellman
- 1685 DSA - Digital Signature Algorithm specified in FIPS 186-3
- 1686 DSKPP - Dynamic Symmetric Key Provisioning Protocol
- 1687 ECB - Electronic Code Book
- 1688 ECDH - Elliptic Curve Diffie-Hellman
- 1689 ECDSA - Elliptic Curve Digital Signature Algorithm specified in ANSX9.62
- 1690 HMAC - Keyed-Hash Message Authentication Code specified in FIPS 198
- 1691 HTTP - Hyper Text Transfer Protocol
- 1692 HTTP(S) - Hyper Text Transfer Protocol (Secure socket)
- 1693 IEEE - Institute of Electrical and Electronics Engineers
- 1694 IETF - Internet Engineering Task Force
- 1695 IPsec - Internet Protocol Security
- 1696 IV - Initialization Vector
- 1697 KMIP - Key Management Interoperability Protocol
- 1698 MAC - Message Authentication Code
- 1699 MD5 - Message Digest 5 Algorithm
- 1700 PBKDF2 - Password-Based Key Derivation Function 2
- 1701 PGP - Pretty Good Privacy
- 1702 PKCS - Public Key Cryptography Standards
- 1703 POSIX - Portable Operating System Interface
- 1704 RFC - Request for Comments documents of IETF
- 1705 RSA - Rivest, Shamir, Adelman (an algorithm)
- 1706 SHA-1 - Secure Hash Algorithm Revision One
- 1707 SSL/TLS - Secure Sockets Layer/Transport Layer Security

- 1708 S/MIME - Secure/Multipurpose Internet Mail Extensions
- 1709 TCP - Transport Control Protocol
- 1710 TTLV - Tag, Type, Length, Value
- 1711 URI - Unique Resource Identifier
- 1712 UTF - Universal Transformation Format
- 1713 XML - Extensible Markup Language

---

1714 **E. Acknowledgements**

1715 The following individuals have participated in the creation of this specification and are gratefully  
1716 acknowledged:

1717 **Original Authors of the initial contribution:**

- 1718 David Babcock, HP
- 1719 Steven Bade, IBM
- 1720 Paolo Bezoari, NetApp
- 1721 Mathias Björkqvist, IBM
- 1722 Bruce Brinson, EMC
- 1723 Christian Cachin, IBM
- 1724 Tony Crossman, Thales/nCipher
- 1725 Stan Feather, HP
- 1726 Indra Fitzgerald, HP
- 1727 Judy Furlong, EMC
- 1728 Jon Geater, Thales/nCipher
- 1729 Bob Griffin, EMC
- 1730 Robert Haas, IBM (editor)
- 1731 Timothy Hahn, IBM
- 1732 Jack Harwood, EMC
- 1733 Walt Hubis, LSI
- 1734 Glen Jaquette, IBM
- 1735 Jeff Kravitz, IBM (editor emeritus)
- 1736 Michael McIntosh, IBM
- 1737 Brian Metzger, HP
- 1738 Anthony Nadalin, IBM
- 1739 Elaine Palmer, IBM
- 1740 Joe Pato, HP
- 1741 René Pawlitzek, IBM
- 1742 Subhash Sankuratipati, NetApp
- 1743 Mark Schiller, HP
- 1744 Martin Skagen, Brocade
- 1745 Marcus Streets, Thales/nCipher
- 1746 John Tattan, EMC
- 1747 Karla Thomas, Brocade
- 1748 Marko Vukolić, IBM
- 1749 Steve Wierenga, HP

1750 **Participants:**

- 1751 TBD



1752

---

## F. Revision History

Revision	Date	Editor	Changes Made
ed-0.98	2009-05-21	Robert Haas	Changes to TTLV format for 64-bit alignment. Appendices indicated as non normative.
ed-0.98	2009-04-24	Robert Haas	Initial conversion of input document to OASIS format together with clarifications.

1753