



KMIP Additional Message Encodings Version 1.0

Committee Specification Draft 01 / Public Review Draft 01

09 January 2014

Specification URIs

This version:

<http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csprd01/kmip-addtl-msg-enc-v1.0-csprd01.doc> (Authoritative)
<http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csprd01/kmip-addtl-msg-enc-v1.0-csprd01.html>
<http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csprd01/kmip-addtl-msg-enc-v1.0-csprd01.pdf>

Previous version:

N/A

Latest version:

<http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/kmip-addtl-msg-enc-v1.0.doc>
(Authoritative)
<http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/kmip-addtl-msg-enc-v1.0.html>
<http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/kmip-addtl-msg-enc-v1.0.pdf>

Technical Committee:

OASIS Key Management Interoperability Protocol (KMIP) TC

Chairs:

Robert Griffin (robert.griffin@rsa.com), EMC Corporation
Subhash Sankuratripati (Subhash.Sankuratripati@netapp.com), NetApp

Editor:

Tim Hudson (tjh@cryptsoft.com), Cryptsoft Pty Ltd.

Related work:

This specification is related to:

- *Key Management Interoperability Protocol Profiles Version 1.0*. 01 October 2010. OASIS Standard. <http://docs.oasis-open.org/kmip/profiles/v1.0/os/kmip-profiles-1.0-os.html>.
- *Key Management Interoperability Protocol Specification Version 1.1*. Latest version. <http://docs.oasis-open.org/kmip/spec/v1.1/kmip-spec-v1.1.html>.
- *Key Management Interoperability Protocol Use Cases Version 1.0*. Latest version. <http://docs.oasis-open.org/kmip/usecases/v1.0/kmip-usecases-1.0.html>.
- *Key Management Interoperability Protocol Usage Guide Version 1.1*. Latest version. <http://docs.oasis-open.org/kmip/ug/v1.1/kmip-ug-v1.1.html>.

Abstract:

Describes additional (optional) message encodings as an alternative to the (mandatory) raw TTLV encoding including:

- HTTP

- JSON
- XML

Status:

This document was last revised or approved by the OASIS Key Management Interoperability Protocol (KMIP) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at <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>).

Citation format:

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

[kmip-addtl-msg-enc-v1.0]

KMIP Additional Message Encodings Version 1.0. Edited by Tim Hudson. 09 January 2014. OASIS Committee Specification Draft 01 / Public Review Draft 01. <http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csprd01/kmip-addtl-msg-enc-v1.0-csprd01.html>. Latest version: <http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/kmip-addtl-msg-enc-v1.0.html>.

Notices

Copyright © OASIS Open 2014. 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 name "OASIS" is a trademark 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/policies-guidelines/trademark> for above guidance.

Table of Contents

1	Introduction.....	5
1.1	Terminology.....	5
1.2	Normative References.....	5
1.3	Non-Normative References.....	6
2	HTTPS Profile.....	7
2.1	Authentication Suite.....	7
2.2	KMIP Port Number.....	7
2.3	Request URI.....	7
2.4	HTTP Encoding.....	7
3	HTTPS Profile Test Cases.....	8
3.1.1	MSGENC-HTTPS-1-10 - Query, Maximum Response Size.....	8
4	JSON Profile.....	13
4.1	JSON Encoding.....	13
4.1.1	Hex representations.....	13
4.1.2	Tags.....	13
4.1.3	Normalizing Names.....	13
4.1.4	Type.....	14
4.1.5	Value.....	14
4.1.6	JSON Object.....	15
5	JSON Profile Test Cases.....	17
5.1.1	MSGENC-JSON-1-10 - Query, Maximum Response Size.....	17
6	XML Profile.....	22
6.1	XML Encoding.....	22
6.1.1	Hex representations.....	22
6.1.2	Tags.....	22
6.1.3	Normalizing Names.....	22
6.1.4	Type.....	23
6.1.5	Value.....	23
6.1.6	XML Element Encoding.....	24
7	XML Profile Test Cases.....	26
7.1.1	MSGENC-XML-1-10 - Query, Maximum Response Size.....	26
8	Conformance.....	29
8.1	HTTPS Profile Conformance.....	29
8.2	JSON Profile Conformance.....	29
8.3	XML Profile Conformance.....	29
8.4	Permitted Test Case Variations.....	29
8.4.1	Variable Items.....	29
8.4.2	Variable behavior.....	31
Appendix A.	Acknowledgments.....	32
Appendix B.	KMIP Specification Cross Reference.....	35
Appendix C.	Revision History.....	40

1 Introduction

For normative definition of the elements of KMIP see the [KMIP Specification](#) [KMIP-SPEC] and the [KMIP Profiles](#) [KMIP-PROF].

Illustrative guidance for the implementation of KMIP clients and servers is provided in the [KMIP Usage Guide](#) [KMIP-UG].

This profile defines the necessary encoding rules for the transport of KMIP TTLV messages encoded in:

- [Hypertext Transfer Protocol](#) [RFC2616] over [TLS](#) as specified in [HTTP over TLS](#) [RFC2818]
- [JavaScript Object Notification](#) [RFC4627]
- [Extensible Markup Language](#) [XML]

1.1 Terminology

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

1.2 Normative References

- [RFC2119] Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
- [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- [RFC2246] T. Dierks and C. Allen, *The TLS Protocol, Version 1.0*, IETF RFC 2246, Jan 1999, <http://www.ietf.org/rfc/rfc2246.txt>
- [RFC2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *Hypertext Transfer Protocol -- HTTP/1.1*, <http://www.ietf.org/rfc/rfc2616.txt>, IETF RFC 2616, June 1999.
- [RFC2818] E. Rescorla, *HTTP over TLS*, IETF RFC 2818, May 2000, <http://www.ietf.org/rfc/rfc2818.txt>
- [RFC4627] D. Crockford, *The application/json Media Type for JavaScript Object Notation (JSON)* July 2006, [http:// tools.ietf.org/html/rfc4627](http://tools.ietf.org/html/rfc4627)
- [XML] Bray, Tim, et.al. eds, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, 198 W3C Recommendation 26 November 2008, available at <http://www.w3.org/TR/2008/REC-xml-20081126/>
- [KMIP-SPEC] One or more of [KMIP-SPEC-1_0], [KMIP-SPEC-1_1], [KMIP-SPEC-1_2]
- [KMIP-SPEC-1_0] Key Management Interoperability Protocol Specification Version 1.0 <http://docs.oasis-open.org/kmip/spec/v1.0/os/kmip-spec-1.0-os.doc> OASIS Standard, October 2010.
- [KMIP-SPEC-1_1] *Key Management Interoperability Protocol Specification Version 1.1*. <http://docs.oasis-open.org/kmip/spec/v1.1/os/kmip-spec-v1.1-os.doc> OASIS Standard. 24 January 2013.
- [KMIP-SPEC-1_2] *Key Management Interoperability Protocol Specification Version 1.2*. [URL](#) Candidate OASIS Standard 01. **DD MMM YYYY**.
- [KMIP-PROF] One or more of [KMIP-PROF-1_0], [KMIP-PROF-1_1], [KMIP-PROF-1_2]
- [KMIP-PROF-1_0] *Key Management Interoperability Protocol Usage Guide Version 1.0*. <http://docs.oasis-open.org/kmip/profiles/v1.0/os/kmip-profiles-1.0-os.doc> OASIS Standard. 1 October 2010.

- 45 **[KMIP-PROF-1_1]** *Key Management Interoperability Protocol Usage Guide Version 1.1.*
46 <http://docs.oasis-open.org/kmip/profiles/v1.1/os/kmip-profiles-v1.1-os.doc>
47 OASIS Standard 01. 24 January 2013.
48 **[KMIP-PROF-1_2]** *Key Management Interoperability Protocol Usage Guide Version 1.2.*
49 URL
50 Candidate OASIS Standard 01. DD MMM YYYY.

51 **1.3 Non-Normative References**

- 52 **[KMIP-UG-1_0]** *Key Management Interoperability Protocol Usage Guide Version 1.0.*
53 <http://docs.oasis-open.org/kmip/ug/v1.1/kmip-ug-v1.1-cnd01.doc>
54 Committee Note Draft, 1 December 2011
55 **[KMIP-UG-1_1]** *Key Management Interoperability Protocol Usage Guide Version 1.1.*
56 <http://docs.oasis-open.org/kmip/ug/v1.1/cn01/kmip-ug-v1.1-cn01.doc>
57 Committee Note 01, 27 July 2012
58 **[KMIP-UG-1_2]** *Key Management Interoperability Protocol Usage Guide Version 1.2.*
59 URL
60 Committee Note Draft, DD MMM YYYY
61 **[KMIP-TC-1_1]** *Key Management Interoperability Protocol Test Cases Version 1.1.*
62 [http://docs.oasis-open.org/kmip/testcases/v1.1/cn01/kmip-testcases-v1.1-](http://docs.oasis-open.org/kmip/testcases/v1.1/cn01/kmip-testcases-v1.1-cn01.doc)
63 [cn01.doc](http://docs.oasis-open.org/kmip/testcases/v1.1/cn01/kmip-testcases-v1.1-cn01.doc), Committee Note 01, 27 July 2012.
64 **[KMIP-TC-1_2]** *Key Management Interoperability Protocol Test Cases Version 1.2.*
65 URL, Committee Note Draft, DD MMM YYYY.
66 **[KMIP-UC]** *Key Management Interoperability Protocol Use Cases Version 1.0.*
67 [http://docs.oasis-open.org/kmip/usecases/v1.0/cs01/kmip-usecases-1.0-cs-](http://docs.oasis-open.org/kmip/usecases/v1.0/cs01/kmip-usecases-1.0-cs-01.doc)
68 [01.doc](http://docs.oasis-open.org/kmip/usecases/v1.0/cs01/kmip-usecases-1.0-cs-01.doc), Committee Specification, 15 June 2010.
69
70

71 2 HTTPS Profile

72 The Hypertext Transfer Protocol over Transport Layer Security (HTTPS) is simply the use of HTTP over
73 TLS in the same manner that HTTP is used over TCP.

74 KMIP over HTTPS is simply the use of KMIP messages over HTTPS in the same manner that KMIP is
75 used over TLS.

76 2.1 Authentication Suite

77 Implementations conformant to this profile SHALL support one or more of the Authentication Suites
78 defined within section 3 of [KMIP-PROF]. The establishment of the trust relationship between the KMIP
79 client and the KMIP server is the same as the defined base profiles.

80 2.2 KMIP Port Number

81 KMIP servers conformant to this profile MAY use TCP port number 5696, as assigned by IANA, to receive
82 and send KMIP messages provided that both HTTP and non-HTTP encoded messages are supported.

83 KMIP clients SHALL enable end user configuration of the TCP port number used, as a KMIP server may
84 specify a different TCP port number.

85 2.3 Request URI

86 KMIP servers conformant to this profile SHOULD support the value */kmip* as the target URI.

87 KMIP clients SHALL enable end user configuration of the target URI used as a KMIP server may specify
88 a different target URI.

89 2.4 HTTP Encoding

90 KMIP client implementations conformant to this profile:

- 91 1. SHALL support HTTP/1.0 and/or HTTP/1.1 over TLS conformant to [RFC2818]
- 92 2. SHALL use the POST request method
- 93 3. SHALL specify a Content-Type of "application/octet-stream"
- 94 4. SHALL specify a Content-Length
- 95 5. SHALL specify a Cache-Control of "no-cache"
- 96 6. SHALL send KMIP TTLV message in binary format as the body of the HTTP request

97

98 KMIP server implementations conformant to this profile:

- 99 1. SHALL support HTTP/1.0 and HTTP/1.1 over TLS conformant to [RFC2818]
- 100 2. SHALL return HTTP response code 200 if a KMIP response is available
- 101 3. SHALL specify a Content-Type of "application/octet-stream"
- 102 4. SHALL specify a Content-Length
- 103 5. SHALL specify a Cache-Control of "no-cache"
- 104 6. SHALL send KMIP TTLV message in binary format as the body of the HTTP request

105

106 KMIP servers that support server to client operations SHALL behave as an HTTPS client. KMIP clients
107 that support responding to server to client operations SHALL behave as a HTTPS server.

108

109 3 HTTPS Profile Test Cases

110 This section contains a test case that demonstrates the HTTPS profile encoding using test case 12.1 from
 111 [KMIP-TC] using protocol version 1.0 which exercises the Query operation and the Maximum Response
 112 Size header field.

113 3.1.1 MSGENC-HTTPS-1-10 - Query, Maximum Response Size

114 Perform a Query operation, querying the Operations and Objects supported by the server, with a
 115 restriction on the Maximum Response Size set in the request header. Since the resulting Query response
 116 is too big, an error is returned. Increase the Maximum Response Size, resubmit the Query request, and
 117 get a successful response.

118 The specific list of operations and object types returned in the response MAY vary.

```

# TIME 0
0001 <RequestMessage>
0002   <RequestHeader>
0003     <ProtocolVersion>
0004       <ProtocolVersionMajor type="Integer" value="1"/>
0005       <ProtocolVersionMinor type="Integer" value="0"/>
0006     </ProtocolVersion>
0007     <MaximumResponseSize type="Integer" value="256"/>
0008     <BatchCount type="Integer" value="1"/>
0009   </RequestHeader>
0010   <BatchItem>
0011     <Operation type="Enumeration" value="Query"/>
0012     <RequestPayload>
0013       <QueryFunction type="Enumeration" value="QueryOperations"/>
0014       <QueryFunction type="Enumeration" value="QueryObjects"/>
0015     </RequestPayload>
0016   </BatchItem>
0017 </RequestMessage>

42007801000000904200770100000048420069010000002042006a02000000040000000100000000
42006b02000000040000000000000000420050020000000400000100000000004200d0200000004
000000010000000042000f010000003842005c050000000400000018000000004200790100000020
4200740500000004000000010000000042007405000000040000000200000000

00000000: 50 4f 53 54 20 2f 6b 6d-69 70 20 48 54 54 50 2f      POST /kmip HTTP/
00000010: 31 2e 30 0d 0a 50 72 61-67 6d 61 3a 20 6e 6f 2d      1.0..Pragma: no-
00000020: 63 61 63 68 65 0d 0a 43-61 63 68 65 2d 43 6f 6e      cache..Cache-Con
00000030: 74 72 6f 6c 3a 20 6e 6f-2d 63 61 63 68 65 0d 0a      trol: no-cache..
00000040: 43 6f 6e 6e 65 63 74 69-6f 6e 3a 20 6b 65 65 70      Connection: keep
00000050: 2d 61 6c 69 76 65 0d 0a-43 6f 6e 74 65 6e 74 2d      -alive..Content-
00000060: 54 79 70 65 3a 20 61 70-70 6c 69 63 61 74 69 6f      Type: applicatio
00000070: 6e 2f 6f 63 74 65 74 2d-73 74 72 65 61 6d 0d 0a      n/octet-stream..
00000080: 43 6f 6e 74 65 6e 74 2d-4c 65 6e 67 74 68 3a 20      Content-Length:
00000090: 31 35 32 20 20 20 20 20-20 20 0d 0a 0d 0a 42 00      152      ....B.
000000a0: 15 32 78 01 00 00 00 90-42 00 77 01 00 00 00 48      .2x.....B.w....H
000000b0: 42 00 69 01 00 00 00 20-42 00 6a 02 00 00 00 04      B.i.... B.j.....
000000c0: 00 00 00 01 00 00 00 00-42 00 6b 02 00 00 00 04      .....B.k.....
000000d0: 00 00 00 00 00 00 00 00-42 00 50 02 00 00 00 04      .....B.P.....
000000e0: 00 00 01 00 00 00 00 00-42 00 0d 02 00 00 00 04      .....B.....
000000f0: 00 00 00 01 00 00 00 00-42 00 0f 01 00 00 00 38      .....B.....8
00000100: 42 00 5c 05 00 00 00 04-00 00 18 00 00 00 00      B.\.....
00000110: 42 00 79 01 00 00 00 20-42 00 74 05 00 00 00 04      B.y.... B.t.....
00000120: 00 00 00 01 00 00 00 00-42 00 74 05 00 00 00 04      .....B.t.....
00000130: 00 00 00 02 00 00 00 00-
0018 <ResponseMessage>
0019   <ResponseHeader>

```


0020	<ProtocolVersion>
0021	<ProtocolVersionMajor type="Integer" value="1"/>
0022	<ProtocolVersionMinor type="Integer" value="0"/>
0023	</ProtocolVersion>
0024	<TimeStamp type="DateTime" value="2013-06-26T09:09:17+00:00"/>
0025	<BatchCount type="Integer" value="1"/>
0026	</ResponseHeader> <BatchItem>
0027	<Operation type="Enumeration" value="Query"/>
0028	<ResultStatus type="Enumeration" value="OperationFailed"/>
0029	<ResultReason type="Enumeration" value="ResponseTooLarge"/>
0030	<ResultMessage type="TextString" value="TOO_LARGE"/>
0031	</BatchItem>
0032	</ResponseMessage>
	<pre> 42007b01000000a042007a0100000048420069010000002042006a02000000040000000100000000 42006b0200000004000000000000000042009209000000080000000051caafbd42000d0200000004 0000000100000000042000f010000004842005c0500000004000000180000000042007f0500000004 000000010000000042007e0500000004000000020000000042007d0700000009544f4f5f4c415247 4500000000000000 00000000: 48 54 54 50 2f 31 2e 31-20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK. 00000010: 0a 43 6f 6e 74 65 6e 74-2d 54 79 70 65 3a 20 61 .Content-Type: a 00000020: 70 70 6c 69 63 61 74 69-6f 6e 2f 6f 63 74 65 74 pplication/octet 00000030: 2d 73 74 72 65 61 6d 0d-0a 43 6f 6e 74 65 6e 74 -stream..Content 00000040: 2d 4c 65 6e 67 74 68 3a-20 31 36 38 0d 0a 0d 0a -Length: 168.... 00000050: 42 00 7b 01 00 00 00 a0-42 00 7a 01 00 00 00 48 B.{... B.z....H 00000060: 42 00 69 01 00 00 00 20-42 00 6a 02 00 00 00 04 B.i.... B.j.... 00000070: 00 00 00 01 00 00 00 00-42 00 6b 02 00 00 00 04B.k.... 00000080: 00 00 00 00 00 00 00 00-42 00 92 09 00 00 00 08B..... 00000090: 00 00 00 00 51 ca af bd-42 00 0d 02 00 00 00 04QJ/=B..... 000000a0: 00 00 00 01 00 00 00 00-42 00 0f 01 00 00 00 48B.....H 000000b0: 42 00 5c 05 00 00 00 04-00 00 18 00 00 00 00 B.\..... 000000c0: 42 00 7f 05 00 00 00 04-00 00 01 00 00 00 00 B..... 000000d0: 42 00 7e 05 00 00 00 04-00 00 02 00 00 00 00 B.~..... 000000e0: 42 00 7d 07 00 00 00 09-54 4f 4f 5f 4c 41 52 47 B.}.....TOO_LARG 000000f0: 45 00 00 00 00 00 00 00- E..... </pre>
	# TIME 1
0032	<RequestMessage>
0033	<RequestHeader>
0034	<ProtocolVersion>
0035	<ProtocolVersionMajor type="Integer" value="1"/>
0036	<ProtocolVersionMinor type="Integer" value="0"/>
0037	</ProtocolVersion>
0038	<MaximumResponseSize type="Integer" value="2048"/>
0039	<BatchCount type="Integer" value="1"/>
0040	</RequestHeader>
0041	<BatchItem>
0042	<Operation type="Enumeration" value="Query"/>
0043	<RequestPayload>
0044	<QueryFunction type="Enumeration" value="QueryOperations"/>
0045	<QueryFunction type="Enumeration" value="QueryObjects"/>
0046	</RequestPayload>
0047	</BatchItem>
0048	</RequestMessage>
	<pre> 42007801000000904200770100000048420069010000002042006a02000000040000000100000000 42006b02000000040000000000000000420050020000000400000800000000042000d0200000004 000000010000000042000f010000003842005c050000000400000018000000004200790100000020 4200740500000004000000010000000042007405000000040000000200000000 00000000: 50 4f 53 54 20 2f 6b 6d-69 70 20 48 54 54 50 2f POST /kmp HTTP/ 00000010: 31 2e 30 0d 0a 50 72 61-67 6d 61 3a 20 6e 6f 2d 1.0..Pragma: no- 00000020: 63 61 63 68 65 0d 0a 43-61 63 68 65 2d 43 6f 6e cache..Cache-Con 00000030: 74 72 6f 6c 3a 20 6e 6f-2d 63 61 63 68 65 0d 0a trol: no-cache.. </pre>

00000040:	43 6f 6e 6e 65 63 74 69-6f 6e 3a 20 6b 65 65 70	Connection: keep
00000050:	2d 61 6c 69 76 65 0d 0a-43 6f 6e 74 65 6e 74 2d	-alive..Content-
00000060:	54 79 70 65 3a 20 61 70-70 6c 69 63 61 74 69 6f	Type: applicatio
00000070:	6e 2f 6f 63 74 65 74 2d-73 74 72 65 61 6d 0d 0a	n/octet-stream..
00000080:	43 6f 6e 74 65 6e 74 2d-4c 65 6e 67 74 68 3a 20	Content-Length:
00000090:	31 35 32 20 20 20 20 20-20 20 0d 0a 0d 0a 42 00	152B.
000000a0:	15 32 78 01 00 00 00 90-42 00 77 01 00 00 00 48	.2x.....B.w....H
000000b0:	42 00 69 01 00 00 00 20-42 00 6a 02 00 00 00 04	B.i..... B.j.....
000000c0:	00 00 00 01 00 00 00 00-42 00 6b 02 00 00 00 04B.k.....
000000d0:	00 00 00 00 00 00 00 00-42 00 50 02 00 00 00 04B.P.....
000000e0:	00 00 08 00 00 00 00 00-42 00 0d 02 00 00 00 04B.....
000000f0:	00 00 00 01 00 00 00 00-42 00 0f 01 00 00 00 38B.....8
00000100:	42 00 5c 05 00 00 00 04-00 00 00 18 00 00 00 00	B.\.....
00000110:	42 00 79 01 00 00 00 20-42 00 74 05 00 00 00 04	B.y..... B.t.....
00000120:	00 00 00 01 00 00 00 00-42 00 74 05 00 00 00 04B.t.....
00000130:	00 00 00 02 00 00 00 00-

```

0049 <ResponseMessage>
0050   <ResponseHeader>
0051     <ProtocolVersion>
0052       <ProtocolVersionMajor type="Integer" value="1"/>
0053       <ProtocolVersionMinor type="Integer" value="0"/>
0054     </ProtocolVersion>
0055     <TimeStamp type="DateTime" value="2013-06-26T09:09:17+00:00"/>
0056     <BatchCount type="Integer" value="1"/>
0057   </ResponseHeader>
0058   <BatchItem>
0059     <Operation type="Enumeration" value="Query"/>
0060     <ResultStatus type="Enumeration" value="Success"/>
0061     <ResponsePayload>
0062       <Operation type="Enumeration" value="Query"/>
0063       <Operation type="Enumeration" value="Locate"/>
0064       <Operation type="Enumeration" value="Destroy"/>
0065       <Operation type="Enumeration" value="Get"/>
0066       <Operation type="Enumeration" value="Create"/>
0067       <Operation type="Enumeration" value="Register"/>
0068       <Operation type="Enumeration" value="GetAttributes"/>
0069       <Operation type="Enumeration" value="GetAttributeList"/>
0070       <Operation type="Enumeration" value="AddAttribute"/>
0071       <Operation type="Enumeration" value="ModifyAttribute"/>
0072       <Operation type="Enumeration" value="DeleteAttribute"/>
0073       <Operation type="Enumeration" value="Activate"/>
0074       <Operation type="Enumeration" value="Revoke"/>
0075       <Operation type="Enumeration" value="Poll"/>
0076       <Operation type="Enumeration" value="Cancel"/>
0077       <Operation type="Enumeration" value="Check"/>
0078       <Operation type="Enumeration" value="GetUsageAllocation"/>
0079       <Operation type="Enumeration" value="CreateKeyPair"/>
0080       <Operation type="Enumeration" value="ReKey"/>
0081       <Operation type="Enumeration" value="Archive"/>
0082       <Operation type="Enumeration" value="Recover"/>
0083       <Operation type="Enumeration" value="ObtainLease"/>
0084       <Operation type="Enumeration" value="Certify"/>
0085       <Operation type="Enumeration" value="ReCertify"/>
0086       <Operation type="Enumeration" value="Notify"/>
0087       <Operation type="Enumeration" value="Put"/>
0088       <ObjectType type="Enumeration" value="Certificate"/>
0089       <ObjectType type="Enumeration" value="SymmetricKey"/>
0090       <ObjectType type="Enumeration" value="SecretData"/>
0091       <ObjectType type="Enumeration" value="PublicKey"/>
0092       <ObjectType type="Enumeration" value="PrivateKey"/>
0093       <ObjectType type="Enumeration" value="Template"/>

```

```

0094 <ObjectType type="Enumeration" value="OpaqueObject"/>
0095 <ObjectType type="Enumeration" value="SplitKey"/>
0096 </ResponsePayload>
0097 </BatchItem>
0098 </ResponseMessage>

42007b01000002a042007a0100000048420069010000002042006a02000000040000000100000000
42006b0200000004000000000000000042009209000000080000000051caafbd42000d0200000004
000000010000000042000f010000024842005c0500000004000000180000000042007f0500000004
000000000000000042007c010000022042005c0500000004000000180000000042005c0500000004
000000080000000042005c0500000004000000140000000042005c05000000040000000a00000000
42005c0500000004000000010000000042005c0500000004000000030000000042005c0500000004
0000000b0000000042005c05000000040000000c0000000042005c05000000040000000d00000000
42005c05000000040000000e0000000042005c05000000040000000f0000000042005c0500000004
000000120000000042005c0500000004000000130000000042005c05000000040000001a00000000
42005c0500000004000000190000000042005c050000000400000004000000090000000042005c0500000004
000000110000000042005c0500000004000000020000000042005c05000000040000000400000000
42005c0500000004000000150000000042005c0500000004000000160000000042005c0500000004
000000100000000042005c0500000004000000060000000042005c05000000040000000700000000
42005c05000000040000001b0000000042005c05000000040000001c0000000042005c0500000004
00000001000000004200570500000004000000020000000042005705000000040000000700000000
42005705000000040000000300000000420057050000000400000004000000004200570500000004
00000006000000004200570500000004000000080000000042005705000000040000000500000000

00000000: 48 54 54 50 2f 31 2e 31-20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.
00000010: 0a 43 6f 6e 74 65 6e 74-2d 54 79 70 65 3a 20 61 .Content-Type: a
00000020: 70 70 6c 69 63 61 74 69-6f 6e 2f 6f 63 74 65 74 pplication/octet
00000030: 2d 73 74 72 65 61 6d 0d-0a 43 6f 6e 74 65 6e 74 -stream..Content
00000040: 2d 4c 65 6e 67 74 68 3a-20 36 38 30 0d 0a 0d 0a -Length: 680....
00000050: 42 00 7b 01 00 00 02 a0-42 00 7a 01 00 00 00 48 B.{.... B.z....H
00000060: 42 00 69 01 00 00 00 20-42 00 6a 02 00 00 00 04 B.i.... B.j.....
00000070: 00 00 00 01 00 00 00 00-42 00 6b 02 00 00 00 04 .....B.k.....
00000080: 00 00 00 00 00 00 00 00-42 00 92 09 00 00 00 08 .....B.....
00000090: 00 00 00 00 51 ca af bd-42 00 0d 02 00 00 00 04 ...QJ/=B.....
000000a0: 00 00 00 01 00 00 00 00-42 00 0f 01 00 00 02 48 .....B.....H
000000b0: 42 00 5c 05 00 00 00 04-00 00 18 00 00 00 00 00 B.\.....
000000c0: 42 00 7f 05 00 00 00 04-00 00 00 00 00 00 00 00 B.....
000000d0: 42 00 7c 01 00 00 02 20-42 00 5c 05 00 00 00 04 B.|.... B.\.....
000000e0: 00 00 00 18 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
000000f0: 00 00 00 08 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000100: 00 00 00 14 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000110: 00 00 00 0a 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000120: 00 00 00 01 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000130: 00 00 00 03 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000140: 00 00 00 0b 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000150: 00 00 00 0c 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000160: 00 00 00 0d 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000170: 00 00 00 0e 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000180: 00 00 00 0f 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000190: 00 00 00 12 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
000001a0: 00 00 00 13 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
000001b0: 00 00 00 1a 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
000001c0: 00 00 00 19 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
000001d0: 00 00 00 09 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
000001e0: 00 00 00 11 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
000001f0: 00 00 00 02 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000200: 00 00 00 04 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000210: 00 00 00 15 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000220: 00 00 00 16 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000230: 00 00 00 10 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000240: 00 00 00 06 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000250: 00 00 00 07 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000260: 00 00 00 1b 00 00 00 00-42 00 5c 05 00 00 00 04 .....B.\.....
00000270: 00 00 00 1c 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....
00000280: 00 00 00 01 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....
00000290: 00 00 00 02 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....
000002a0: 00 00 00 07 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....
000002b0: 00 00 00 03 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....
000002c0: 00 00 00 04 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....
000002d0: 00 00 00 06 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....
000002e0: 00 00 00 08 00 00 00 00-42 00 57 05 00 00 00 04 .....B.W.....

```

000002f0: 00 00 00 05 00 00 00 00-
------------------------------------	-------

119

120 4 JSON Profile

121 The JSON profile specifies the use of KMIP replacing the TTLV message encoding with a JSON message
122 encoding.

123 4.1 JSON Encoding

124 4.1.1 Hex representations

125 Hex representations of numbers must always begin with '0x' and must not include any spaces. They may
126 use either upper or lower case 'a'-'f'. The hex representation must include all leading zeros or sign
127 extension bits when representing a value of a fixed width such as Tags (3 bytes), Integer (32-bit signed
128 big-endian), Long Integer (64-bit signed big-endian) and Big Integer (big-endian multiple of 8 bytes). The
129 Integer values for -1, 0, 1 are represented as "0xffffffff", "0x00000000", "0x00000001". Hex
130 representation for Byte Strings are similar to numbers, but do not include the '0x' prefix, and can be of
131 any length.

132 4.1.2 Tags

133 Tags are a String that may contain either:

- 134 • The 3-byte tag hex value prefixed with '0x'
- 135 • The normalised text of a Tag as specified in the KMIP Specification

136 Other text values may be used such as published names of Extension tags, or names of new tags added
137 in future KMIP versions. Producers may however choose to use hex values for these tags to ensure they
138 are understood by all consumers.

139 4.1.3 Normalizing Names

140 KMIP text values of Tags, Types and Enumerations SHALL be normalized to create a 'CamelCase'
141 format that would be suitable to be used as a variable name in C/Java or an JSON name.

142 The basic approach to converting from KMIP text to CamelCase is to separate the text into individual
143 word tokens (rules 1-4), capitalize the first letter of each word (rule 5) and then join with spaces removed
144 (rule 6). The tokenizing splits on whitespace and on dashes where the token following is a valid word.
145 The tokenizing also removes round brackets and shifts decimals from the front to the back of the first
146 word in each string. The following rules SHALL be applied to create the normalized CamelCase form:

- 147 1. Replace round brackets '(' , ')' with spaces
 - 148 2. If a non-word char (not alpha, digit or underscore) is followed by a letter (either upper or lower
149 case) then a lower case letter, replace the non-word char with space
 - 150 3. Replace remaining non-word chars (except whitespace) with underscore.
 - 151 4. If the first word begins with a digit, move all digits at start of first word to end of first word
 - 152 5. Capitalize the first letter of each word
 - 153 6. Concatenate all words with spaces removed
- 154

```

155 # 1. Replace brackets with space
156 noBrackets = re.sub('[()]', ' ', enumName)
157 # 2. replace \W with space if followed by letter, lower
158 nonWordToSpace = re.sub('\W([A-Za-z][a-z])', r' \1', noBrackets)
159 # 3. non-word to underscore
160 words = [re.sub('\W', '_', s) for s in nonWordToSpace.split()]
161 # 4. move numbers to end of first word
162 words[0] = re.sub('^\d+ (.*)', r'\2\1', words[0])
163 # 5. captialize first letter of each word
164 words = [re.sub('^.', s[0].upper(), s) for s in words]
165 # 6. concatenate
166 enumNameCamel = ''.join(words)

```

167 *Example python name normalization code*

```

168
169 # 1. Replace brackets with space
170 $enumName=~s/[()\ ]/ /g;
171 # 2. replace \W with space if followed by letter, lower
172 $enumName=~s/\W([A-Za-z][a-z])/ \1/g;
173 # 3. non-word to underscore
174 @words=split(/ /,$enumName);
175 for($i=0;$i<=$#words;$i++) { $words[$i]=~s/\W/_/g; }
176 # 4. move numbers to end of first word
177 $words[0] =~ s/^\d+ (.*)/\2\1/;
178 # 5. captialize first letter of each word
179 for($i=0;$i<=$#words;$i++) {
180     substr($words[$i],0,1)=~tr/a-z/A-Z/;
181 }
182 # 6. concatenate
183 $enumNameCamel = join(' ',@words);
184

```

185 *Example perl name normalization code*

186 4.1.4 Type

187 Type must be a String containing one of the normalized CamelCase values as defined in the KMIP
188 specification.

- 189 • Structure
- 190 • Integer
- 191 • LongInteger
- 192 • BigInteger
- 193 • Enumeration
- 194 • Boolean
- 195 • TextString
- 196 • ByteString
- 197 • DateTime
- 198 • Interval

199 If type is not included, the default type of Structure SHALL be used.

200 4.1.5 Value

201 The specification of a value is represented differently for each TTLV type.

202 4.1.6 JSON Object

203 For JSON encoding, each TTLV is represented as a JSON Object with properties 'tag', optional
204 'name', 'type' and 'value'.

```
205 {"tag": "ActivationDate", "type": "DateTime", "value": "2001-01-01T10:00:00+10:00"}  
206 {"tag": "0x54FFFF", "name": "SomeExtension", "type": "Integer", "value": "0x00000001"}
```

207 The 'type' property / attribute SHALL have a default value of 'Structure' and may be omitted for
208 Structures.

209 4.1.6.1 Tags

210 Tags are a String that may contain either:

- 211 • The 3-byte tag hex value prefixed with '0x'
- 212 • The normalised text of a Tag as specified in the KMIP Specification

213 Other text values may be used such as published names of Extension tags, or names of new tags added
214 in future KMIP versions. Producers may however choose to use hex values for these tags to ensure they
215 are understood by all consumers.

```
216 {"tag": "0x420001", "type": "DateTime", "value": "2001-01-01T10:00:00+10:00"}  
217 {"tag": "ActivationDate", "type": "DateTime", "value": "2001-01-01T10:00:00+10:00"}  
218 {"tag": "IVCounterNonce", "type": "ByteString", "value": "alb2c3d4"}  
219 {"tag": "PrivateKeyTemplateAttribute", "type": "Structure", "value": []}  
220 {"tag": "0x545352", "type": "TextString", "value": "This is an extension"}  
221 {"tag": "WELL_KNOWN_EXTENSION", "type": "TextString", "value": "This is an extension"}
```

222 4.1.6.2 Structure

223 For JSON, value is an Array containing sub-items, or may be null.

```
224 {"tag": "ProtocolVersion", "type": "Structure", "value": [  
225   {"tag": "ProtocolVersionMajor", "type": "Integer", "value": 1},  
226   {"tag": "ProtocolVersionMajor", "type": "Integer", "value": 0}  
227 ]}  
228 {"tag": "ProtocolVersion", "value": [  
229   {"tag": "ProtocolVersionMajor", "type": "Integer", "value": 1},  
230   {"tag": "ProtocolVersionMajor", "type": "Integer", "value": 0}  
231 ]}
```

232 The 'type' property / attribute is optional for a Structure.

233 4.1.6.3 Integer

234 For JSON, value is either a Number or a hex string.

```
235 {"tag": "BatchCount", "type": "Integer", "value": 10}  
236 {"tag": "BatchCount", "type": "Integer", "value": "0x0000000A"}
```

237 4.1.6.4 Integer - Special case for Masks

238 (Cryptographic Usage Mask, Storage Status Mask):

239 Integer mask values can also be encoded as a String containing mask components. JSON uses '|' as the
240 separator. Components may be either the text of the enumeration value as defined in the KMIP
241 Specification or a 32-bit unsigned big-endian hex string.

```
242 {"tag": "CryptographicUsageMask", "type": "Integer", "value": "0x0000100c"}  
243 {"tag": "CryptographicUsageMask", "type": "Integer", "value": "Encrypt|Decrypt|CertificateSign"}  
244 {"tag": "CryptographicUsageMask", "type": "Integer", "value":  
245 "CertificateSign|0x00000004|0x00000008"}  
246 {"tag": "CryptographicUsageMask", "type": "Integer", "value": "CertificateSign|0x0000000c"}
```

247 **4.1.6.5 Long Integer**

248 For JSON, value is either a Number or a hex string. Note that JS Numbers are 64-bit floating point and
249 can only represent 53-bits of precision, so any values $\geq 2^{52}$ must be represented as hex strings.

```
250 {"tag": "0x540001", "type": "LongInteger", "value": "0xfffffffffffffffffe"}  
251 {"tag": "0x540001", "type": "LongInteger", "value": -2}  
252 {"tag": "UsageLimitsCount", "type": "LongInteger", "value": "0x1000000000000000"}
```

253 Note that this value (2^{60}) is too large to be represented as a Number in JSON.

254 **4.1.6.6 Big Integer**

255 For JSON, value is either a Number or a hex string. Note that Big Integers must be sign extended to
256 contain a multiple of 8 bytes, and as per LongInteger, JS numbers only support a limited range of values.

```
257 {"tag": "X", "type": "BigInteger", "value": 0}  
258 {"tag": "X", "type": "BigInteger", "value": "0x0000000000000000"}
```

259 **4.1.6.7 Enumeration**

260 For JSON, value may contain:

- 261 • Number representing the enumeration 32-bit unsigned big-endian value
- 262 • Hex string representation of 32-bit unsigned big-endian value
- 263 • CamelCase enum text as defined in KMIP 9.1.3.2.x

```
264  
265 {"tag": "0x420057", "type": "Enumeration", "value": 2}  
266 {"tag": "ObjectType", "type": "Enumeration", "value": "0x00000002"}  
267 {"tag": "ObjectType", "type": "Enumeration", "value": "SymmetricKey"}
```

268 **4.1.6.8 Boolean**

269 For JSON, value must be either a hex string, or a JSON Boolean 'true' or 'false'.

```
270 {"tag": "BatchOrderOption", "type": "Boolean", "value": true}  
271 {"tag": "BatchOrderOption", "type": "Boolean", "value": "0x0000000000000001"}
```

272 **4.1.6.9 Text String**

273 For JSON, value must be a String

```
274 {"tag": "AttributeName", "type": "TextString", "value": "Cryptographic Algorithm"}
```

275 **4.1.6.10 Byte String**

276 For JSON, value must be a hex string. Note Byte Strings do not include the '0x' prefix, and do not have
277 any leading bytes.

```
278 {"tag": "MACSignature", "type": "ByteString", "value": "C50F77"}
```

279 **4.1.6.11 Date-Time**

280 For JSON, value must be either a hex string, or an ISO8601 DateTime as used in XSD using format:

```
281 '-'? yyyy '-' mm '-' dd 'T' hh ':' mm ':' ss ('.' s+)? ((( '+' | '-' ) hh ':' mm ) | 'Z')?
```

282 Fractional seconds are not used in KMIP and should not generally be shown. If they are used, they
283 should be ignored (truncated).

```
284 {"tag": "ArchiveDate", "type": "DateTime", "value": "0x000000003a505520"}  
285 {"tag": "ArchiveDate", "type": "DateTime", "value": "2001-01-01T10:00:00+10:00"}
```

286 **4.1.6.12 Interval**

287 For JSON, value is either a Number or a hex string. Note that intervals are 32-bit unsigned big-endian
288 values.

```
289 {"tag": "Offset", "type": "Interval", "value": 27}  
290 {"tag": "Offset", "type": "Interval", "value": "0x0000001b"}
```


291

5 JSON Profile Test Cases

292 This section contains a test case that demonstrates the JSON profile encoding using test case 12.1 from
293 [KMIP-TC] using protocol version 1.0 which exercises the Query operation and the Maximum Response
294 Size header field.

295 5.1.1 MSGENC-JSON-1-10 - Query, Maximum Response Size

296 Perform a Query operation, querying the Operations and Objects supported by the server, with a
297 restriction on the Maximum Response Size set in the request header. Since the resulting Query response
298 is too big, an error is returned. Increase the Maximum Response Size, resubmit the Query request, and
299 get a successful response.

300 The specific list of operations and object types returned in the response MAY vary.

0001	# TIME 0
0002	<RequestMessage>
0003	<RequestHeader>
0004	<ProtocolVersion>
0005	<ProtocolVersionMajor type="Integer" value="1"/>
0006	<ProtocolVersionMinor type="Integer" value="0"/>
0007	</ProtocolVersion>
0008	<MaximumResponseSize type="Integer" value="256"/>
0009	<BatchCount type="Integer" value="1"/>
0010	</RequestHeader>
0011	<BatchItem>
0012	<Operation type="Enumeration" value="Query"/>
0013	<RequestPayload>
0014	<QueryFunction type="Enumeration" value="QueryOperations"/>
0015	<QueryFunction type="Enumeration" value="QueryObjects"/>
0016	</RequestPayload>
0017	</BatchItem>
0017	</RequestMessage>
0017	42007801000000904200770100000048420069010000002042006a02000000040000000100000000 42006b020000000400000000000004200500200000004000001000000000042000d0200000004 000000010000000042000f010000003842005c050000000400000018000000004200790100000020 4200740500000004000000010000000042007405000000040000000200000000
0017	{ "tag": "RequestMessage", "value": [
0017	{ "tag": "RequestHeader", "value": [
0017	{ "tag": "ProtocolVersion", "value": [
0017	{ "tag": "ProtocolVersionMajor", "type": "Integer", "value": "0x00000001" },
0017	{ "tag": "ProtocolVersionMinor", "type": "Integer", "value": "0x00000000" }
0017],
0017	{ "tag": "MaximumResponseSize", "type": "Integer", "value": "0x00000100" },
0017	{ "tag": "BatchCount", "type": "Integer", "value": "0x00000001" }
0017],
0017	{ "tag": "BatchItem", "value": [
0017	{ "tag": "Operation", "type": "Enumeration", "value": "Query" },
0017	{ "tag": "RequestPayload", "value": [
0017	{ "tag": "QueryFunction", "type": "Enumeration", "value": "QueryOperations" },
0017	{ "tag": "QueryFunction", "type": "Enumeration", "value": "QueryObjects" }
0017]]
0017]]
0017]
0018	<ResponseMessage>
0019	<ResponseHeader>
0020	<ProtocolVersion>

<pre> 0021 0022 0023 0024 0025 0026 0027 0028 0029 0030 0031 0032 </pre>	<pre> <ProtocolVersionMajor type="Integer" value="1"/> <ProtocolVersionMinor type="Integer" value="0"/> </ProtocolVersion> <TimeStamp type="DateTime" value="2013-06-26T09:09:17+00:00"/> <BatchCount type="Integer" value="1"/> </ResponseHeader> <BatchItem> <Operation type="Enumeration" value="Query"/> <ResultStatus type="Enumeration" value="OperationFailed"/> <ResultReason type="Enumeration" value="ResponseTooLarge"/> <ResultMessage type="TextString" value="TOO_LARGE"/> </BatchItem> </ResponseMessage> </pre> <p>42007b01000000a042007a0100000048420069010000002042006a02000000040000000100000000 42006b0200000004000000000000000042009209000000080000000051caafbd42000d0200000004 000000010000000042000f010000004842005c0500000004000000180000000042007f0500000004 000000010000000042007e0500000004000000020000000042007d0700000009544f4f5f4c415247 4500000000000000</p> <pre> {"tag":"ResponseMessage", "value":[{"tag":"ResponseHeader", "value":[{"tag":"ProtocolVersion", "value":[{"tag":"ProtocolVersionMajor", "type":"Integer", "value":"0x00000001"}, {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000000"}]}, {"tag":"TimeStamp", "type":"DateTime", "value":"2013-06-26T09:09:17+00:00"}, {"tag":"BatchCount", "type":"Integer", "value":"0x00000001"}]}, {"tag":"BatchItem", "value":[{"tag":"Operation", "type":"Enumeration", "value":"Query"}, {"tag":"ResultStatus", "type":"Enumeration", "value":"OperationFailed"}, {"tag":"ResultReason", "type":"Enumeration", "value":"ResponseTooLarge"}, {"tag":"ResultMessage", "type":"TextString", "value":"TOO_LARGE"}]}]} </pre>
<pre> 0032 0033 0034 0035 0036 0037 0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 </pre>	<pre> # TIME 1 <RequestMessage> <RequestHeader> <ProtocolVersion> <ProtocolVersionMajor type="Integer" value="1"/> <ProtocolVersionMinor type="Integer" value="0"/> </ProtocolVersion> <MaximumResponseSize type="Integer" value="2048"/> <BatchCount type="Integer" value="1"/> </RequestHeader> <BatchItem> <Operation type="Enumeration" value="Query"/> <RequestPayload> <QueryFunction type="Enumeration" value="QueryOperations"/> <QueryFunction type="Enumeration" value="QueryObjects"/> </RequestPayload> </BatchItem> </RequestMessage> </pre> <p>42007801000000904200770100000048420069010000002042006a02000000040000000100000000 42006b020000000400000000000000004200500200000004000008000000000042000d0200000004 000000010000000042000f010000003842005c050000000400000018000000004200790100000020 4200740500000004000000010000000042007405000000040000000200000000</p> <pre> {"tag":"RequestMessage", "value":[{"tag":"RequestHeader", "value":[{"tag":"ProtocolVersion", "value":[</pre>

```
{
  {"tag":"ProtocolVersionMajor", "type":"Integer", "value":"0x00000001"},
  {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000000"}
}],
{"tag":"MaximumResponseSize", "type":"Integer", "value":"0x00000800"},
{"tag":"BatchCount", "type":"Integer", "value":"0x00000001"}
}],
{"tag":"BatchItem", "value":[
  {"tag":"Operation", "type":"Enumeration", "value":"Query"},
  {"tag":"RequestPayload", "value":[
    {"tag":"QueryFunction", "type":"Enumeration", "value":"QueryOperations"},
    {"tag":"QueryFunction", "type":"Enumeration", "value":"QueryObjects"}
  ]}
]}
]]
}}
```

```
0049 <ResponseMessage>
0050 <ResponseHeader>
0051 <ProtocolVersion>
0052 <ProtocolVersionMajor type="Integer" value="1"/>
0053 <ProtocolVersionMinor type="Integer" value="0"/>
0054 </ProtocolVersion>
0055 <TimeStamp type="DateTime" value="2013-06-26T09:09:17+00:00"/>
0056 <BatchCount type="Integer" value="1"/>
0057 </ResponseHeader>
0058 <BatchItem>
0059 <Operation type="Enumeration" value="Query"/>
0060 <ResultStatus type="Enumeration" value="Success"/>
0061 <ResponsePayload>
0062 <Operation type="Enumeration" value="Query"/>
0063 <Operation type="Enumeration" value="Locate"/>
0064 <Operation type="Enumeration" value="Destroy"/>
0065 <Operation type="Enumeration" value="Get"/>
0066 <Operation type="Enumeration" value="Create"/>
0067 <Operation type="Enumeration" value="Register"/>
0068 <Operation type="Enumeration" value="GetAttributes"/>
0069 <Operation type="Enumeration" value="GetAttributeList"/>
0070 <Operation type="Enumeration" value="AddAttribute"/>
0071 <Operation type="Enumeration" value="ModifyAttribute"/>
0072 <Operation type="Enumeration" value="DeleteAttribute"/>
0073 <Operation type="Enumeration" value="Activate"/>
0074 <Operation type="Enumeration" value="Revoke"/>
0075 <Operation type="Enumeration" value="Poll"/>
0076 <Operation type="Enumeration" value="Cancel"/>
0077 <Operation type="Enumeration" value="Check"/>
0078 <Operation type="Enumeration" value="GetUsageAllocation"/>
0079 <Operation type="Enumeration" value="CreateKeyPair"/>
0080 <Operation type="Enumeration" value="ReKey"/>
0081 <Operation type="Enumeration" value="Archive"/>
0082 <Operation type="Enumeration" value="Recover"/>
0083 <Operation type="Enumeration" value="ObtainLease"/>
0084 <Operation type="Enumeration" value="Certify"/>
0085 <Operation type="Enumeration" value="ReCertify"/>
0086 <Operation type="Enumeration" value="Notify"/>
0087 <Operation type="Enumeration" value="Put"/>
0088 <ObjectType type="Enumeration" value="Certificate"/>
0089 <ObjectType type="Enumeration" value="SymmetricKey"/>
0090 <ObjectType type="Enumeration" value="SecretData"/>
0091 <ObjectType type="Enumeration" value="PublicKey"/>
0092 <ObjectType type="Enumeration" value="PrivateKey"/>
0093 <ObjectType type="Enumeration" value="Template"/>
```

```

0094     <ObjectType type="Enumeration" value="OpaqueObject"/>
0095     <ObjectType type="Enumeration" value="SplitKey"/>
0096     </ResponsePayload>
0097 </BatchItem>
0098 </ResponseMessage>

42007b01000002a042007a0100000048420069010000002042006a02000000040000000100000000
42006b0200000004000000000000000042009209000000080000000051caafbd42000d0200000004
000000010000000042000f010000024842005c0500000004000000180000000042007f0500000004
00000000000000042007c010000022042005c0500000004000000180000000042005c0500000004
000000080000000042005c0500000004000000140000000042005c05000000040000000a00000000
42005c0500000004000000010000000042005c0500000004000000030000000042005c0500000004
0000000b0000000042005c05000000040000000c0000000042005c05000000040000000d00000000
42005c05000000040000000e0000000042005c05000000040000000f0000000042005c0500000004
000000120000000042005c0500000004000000130000000042005c05000000040000001a00000000
42005c0500000004000000190000000042005c0500000004000000090000000042005c0500000004
000000110000000042005c0500000004000000020000000042005c05000000040000000400000000
42005c0500000004000000150000000042005c0500000004000000160000000042005c0500000004
000000100000000042005c0500000004000000060000000042005c05000000040000000700000000
42005c05000000040000001b0000000042005c05000000040000001c0000000042005c0500000004
00000001000000004200570500000004000000020000000042005705000000040000000700000000
42005705000000040000000300000000420057050000000400000004000000004200570500000004
00000006000000004200570500000004000000080000000042005705000000040000000500000000

{"tag":"ResponseMessage", "value": [
  {"tag":"ResponseHeader", "value": [
    {"tag":"ProtocolVersion", "value": [
      {"tag":"ProtocolVersionMajor", "type":"Integer", "value":"0x00000001"},
      {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000000"}
    ]},
    {"tag":"TimeStamp", "type":"DateTime", "value":"2013-06-26T09:09:17+00:00"},
    {"tag":"BatchCount", "type":"Integer", "value":"0x00000001"}
  ]},
  {"tag":"BatchItem", "value": [
    {"tag":"Operation", "type":"Enumeration", "value":"Query"},
    {"tag":"ResultStatus", "type":"Enumeration", "value":"Success"},
    {"tag":"ResponsePayload", "value": [
      {"tag":"Operation", "type":"Enumeration", "value":"Query"},
      {"tag":"Operation", "type":"Enumeration", "value":"Locate"},
      {"tag":"Operation", "type":"Enumeration", "value":"Destroy"},
      {"tag":"Operation", "type":"Enumeration", "value":"Get"},
      {"tag":"Operation", "type":"Enumeration", "value":"Create"},
      {"tag":"Operation", "type":"Enumeration", "value":"Register"},
      {"tag":"Operation", "type":"Enumeration", "value":"GetAttributes"},
      {"tag":"Operation", "type":"Enumeration", "value":"GetAttributeList"},
      {"tag":"Operation", "type":"Enumeration", "value":"AddAttribute"},
      {"tag":"Operation", "type":"Enumeration", "value":"ModifyAttribute"},
      {"tag":"Operation", "type":"Enumeration", "value":"DeleteAttribute"},
      {"tag":"Operation", "type":"Enumeration", "value":"Activate"},
      {"tag":"Operation", "type":"Enumeration", "value":"Revoke"},
      {"tag":"Operation", "type":"Enumeration", "value":"Poll"},
      {"tag":"Operation", "type":"Enumeration", "value":"Cancel"},
      {"tag":"Operation", "type":"Enumeration", "value":"Check"},
      {"tag":"Operation", "type":"Enumeration", "value":"GetUsageAllocation"},
      {"tag":"Operation", "type":"Enumeration", "value":"CreateKeyPair"},
      {"tag":"Operation", "type":"Enumeration", "value":"ReKey"},
      {"tag":"Operation", "type":"Enumeration", "value":"Archive"},
      {"tag":"Operation", "type":"Enumeration", "value":"Recover"},
      {"tag":"Operation", "type":"Enumeration", "value":"ObtainLease"},
      {"tag":"Operation", "type":"Enumeration", "value":"Certify"},
      {"tag":"Operation", "type":"Enumeration", "value":"ReCertify"},
      {"tag":"Operation", "type":"Enumeration", "value":"Notify"},
      {"tag":"Operation", "type":"Enumeration", "value":"Put"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"Certificate"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"SymmetricKey"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"SecretData"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"PublicKey"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"PrivateKey"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"Template"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"OpaqueObject"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"SplitKey"}
    ]}
  ]}

```

	}} }} }}
--	----------------

301

302 6 XML Profile

303 The XML profile specifies the use of KMIP replacing the TTLV message encoding with an XML message
304 encoding.

305 6.1 XML Encoding

306 6.1.1 Hex representations

307 Hex representations of numbers must always begin with '0x' and must not include any spaces. They may
308 use either upper or lower case 'a'-'f'. The hex representation must include all leading zeros or sign
309 extension bits when representing a value of a fixed width such as Tags (3 bytes), Integer (32-bit signed
310 big-endian), Long Integer (64-bit signed big-endian) and Big Integer (big-endian multiple of 8 bytes). The
311 Integer values for -1, 0, 1 are represented as "0xffffffff", "0x00000000", "0x00000001". Hex
312 representation for Byte Strings are similar to numbers, but do not include the '0x' prefix, and can be of
313 any length.

314 6.1.2 Tags

315 Tags are a String that may contain either:

- 316 • The 3-byte tag hex value prefixed with '0x'
- 317 • The normalised text of a Tag as specified in the KMIP Specification

318 Other text values may be used such as published names of Extension tags, or names of new tags added
319 in future KMIP versions. Producers may however choose to use hex values for these tags to ensure they
320 are understood by all consumers.

321 6.1.3 Normalizing Names

322 KMIP text values of Tags, Types and Enumerations SHALL be normalized to create a 'CamelCase'
323 format that would be suitable to be used as a variable name in C/Java or an XML element name.

324 The basic approach to converting from KMIP text to CamelCase is to separate the text into individual
325 word tokens (rules 1-4), capitalize the first letter of each word (rule 5) and then join with spaces removed
326 (rule 6). The tokenizing splits on whitespace and on dashes where the token following is a valid word.
327 The tokenizing also removes round brackets and shifts decimals from the front to the back of the first
328 word in each string. The following rules SHALL be applied to create the normalized CamelCase form:

- 329 7. Replace round brackets '(' , ')' with spaces
- 330 8. If a non-word char (not alpha, digit or underscore) is followed by a letter (either upper or lower
331 case) then a lower case letter, replace the non-word char with space
- 332 9. Replace remaining non-word chars (except whitespace) with underscore.
- 333 10. If the first word begins with a digit, move all digits at start of first word to end of first word
- 334 11. Capitalize the first letter of each word
- 335 12. Concatenate all words with spaces removed
- 336

```

337 # 1. Replace brackets with space
338 noBrackets = re.sub('[()]', ' ', enumName)
339 # 2. replace \W with space if followed by letter, lower
340 nonWordToSpace = re.sub('\W([A-Za-z][a-z])', r' \1', noBrackets)
341 # 3. non-word to underscore
342 words = [re.sub('\W', '_', s) for s in nonWordToSpace.split()]
343 # 4. move numbers to end of first word
344 words[0] = re.sub('^\d+ (.*)', r'\2\1', words[0])
345 # 5. captialize first letter of each word
346 words = [re.sub('^.', s[0].upper(), s) for s in words]
347 # 6. concatenate
348 enumNameCamel = ''.join(words)

```

349 *Example python name normalization code*

350

```

351 # 1. Replace brackets with space
352 $enumName=~s/[\\(\\)]/ /g;
353 # 2. replace \W with space if followed by letter, lower
354 $enumName=~s/\W([A-Za-z][a-z])/ \1/g;
355 # 3. non-word to underscore
356 @words=split(/ /,$enumName);
357 for($i=0;$i<=$#words;$i++) { $words[$i]=~s/\W/_/g; }
358 # 4. move numbers to end of first word
359 $words[0] =~ s/^\d+ (.*)/\2\1/;
360 # 5. captialize first letter of each word
361 for($i=0;$i<=$#words;$i++) {
362     substr($words[$i],0,1)=~tr/a-z/A-Z/;
363 }
364 # 6. concatenate
365 $enumNameCamel = join(' ',@words);
366

```

367 *Example perl name normalization code*

368 6.1.4 Type

369 Type must be a String containing one of the normalized CamelCase values as defined in the KMIP
370 specification.

- 371 • Structure
- 372 • Integer
- 373 • LongInteger
- 374 • BigInteger
- 375 • Enumeration
- 376 • Boolean
- 377 • TextString
- 378 • ByteString
- 379 • DateTime
- 380 • Interval

381 If type is not included, the default type of Structure SHALL be used.

382 6.1.5 Value

383 The specification of a value is represented differently for each TTLV type.

384 6.1.6 XML Element Encoding

385 For XML, each TTLV is represented as an XML element with attributes. The general form uses a single
386 element named 'TTLV' with 'tag', optional 'name' and 'type' attributes. This form allows any TTLV
387 including extensions to be encoded. For tags defined in the KMIP Specification or other well-known
388 extensions, a more specific form can be used where each tag is encoded as an element with the same
389 name and includes a 'type' attribute. For either form, structure values are encoded as nested xml
390 elements, and non-structure values are encoded using the 'value' attribute.

```
391  
392 <TTLV tag="0x420001" name="ActivationDate" type="DateTime" value="2001-01-01T10:00:00+10:00"/>  
393 <TTLV tag="0x420001" type="DateTime" value="2001-01-01T10:00:00+10:00"/>  
394 <ActivationDate type="DateTime" value="2001-01-01T10:00:00+10:00"/>  
395 <TTLV tag="0x54FFFF" name="SomeExtension" type="DateTime" value="2001-01-01T10:00:00+10:00"/>  
396
```

397 The 'type' property / attribute SHALL have a default value of 'Structure' and may be omitted for
398 Structures.

399 If namespaces are required, XML elements SHALL use the following namespace:

```
400 urn:oasis:tc:kmip:xmlns
```

401 6.1.6.1 Tags

402 Tags are a String that may contain either:

- 403 • The 3-byte tag hex value prefixed with '0x'
- 404 • The normalised text of a Tag as specified in the KMIP Specification

405 Other text values may be used such as published names of Extension tags, or names of new tags added
406 in future KMIP versions. Producers may however choose to use hex values for these tags to ensure they
407 are understood by all consumers.

```
408 <ActivationDate xmlns="urn:oasis:tc:kmip:xmlns" type="DateTime" value="2001-01-  
409 01T10:00:00+10:00"/>  
410 <IVCounterNonce type="ByteString" value="alb2c3d4"/>  
411 <PrivateKeyTemplateAttribute type="Structure"/>  
412 <TTLV tag="0x545352" name="SomeExtension" type="TextString" value="This is an extension"/>  
413 <WELL_KNOWN_EXTENSION type="TextString" value="This is an extension"/>
```

414 6.1.6.2 Structure

415 For XML, sub-items are nested elements.

```
416 <ProtocolVersion type="Structure">  
417   <ProtocolVersionMajor type="Integer" value="1"/>  
418   <ProtocolVersionMinor type="Integer" value="0"/>  
419 </ProtocolVersion>  
420 <ProtocolVersion>  
421   <ProtocolVersionMajor type="Integer" value="1"/>  
422   <ProtocolVersionMinor type="Integer" value="0"/>  
423 </ProtocolVersion>  
424
```

425 The 'type' property / attribute is optional for a Structure.

426 6.1.6.3 Integer

427 For XML, value is a decimal and uses XML schema type xsd:int

```
428  
429 <BatchCount type="Integer" value="10"/>
```

430 6.1.6.4 Integer - Special case for Masks

431 (Cryptographic Usage Mask, Storage Status Mask):

432 Integer mask values can also be encoded as a String containing mask components. XML uses an
433 attribute with XML type xsd:list which uses a space separator. Components may be either the text of the
434 enumeration value as defined in [KMIP 9.1.3.3.1](#) / [KMIP 9.1.3.3.2](#), or a 32-bit unsigned big-endian hex
435 string.

```
436 <CryptographicUsageMask type="Integer" value="0x0000100c"/>  
437 <CryptographicUsageMask type="Integer" value="Encrypt Decrypt CertificateSign"/>  
438 <CryptographicUsageMask type="Integer" value="CertificateSign 0x00000004 0x00000008"/>  
439 <CryptographicUsageMask type="Integer" value="CertificateSign 0x0000000c"/>
```

440 **6.1.6.5 Long Integer**

441 For XML, value uses XML schema type xsd:long

```
442 <x540001 type="LongInteger" value="-2"/>  
443 <UsageLimitsCount type="LongInteger" value="1152921504606846976"/>
```

444 **6.1.6.6 Big Integer**

445 For XML, value uses XML schema type xsd:hexBinary

```
446 <X type="BigInteger" value="0000000000000000"/>
```

447 **6.1.6.7 Enumeration**

448 For XML, value uses XML schema type xsd:string and is either a hex string or the CamelCase enum text.
449 If an XSD with xsd:enumeration restriction is used to define valid values (as is the case with the XSD
450 included as an appendix), parsers should also accept any hex string in addition to defined enum values.

```
451 <ObjectType type="Enumeration" value="0x00000002"/>  
452 <ObjectType type="Enumeration" value="SymmetricKey"/>
```

453 **6.1.6.8 Boolean**

454 For XML, value uses XML schema type xsd:Boolean

```
455 <BatchOrderOption type="Boolean" value="true"/>
```

456 **6.1.6.9 Text String**

457 XML uses schema type xsd:string

```
458 <AttributeName type="TextString" value="Cryptographic Algorithm"/>
```

459 **6.1.6.10 Byte String**

460 XML uses schema type xsd:hexBinary

```
461 <MACSignature type="ByteString" value="C50F77"/>
```

462 **6.1.6.11 Date-Time**

463 For XML, value uses schema type xsd:dateTime

```
464 <ArchiveDate type="DateTime" value="2001-01-01T10:00:00+10:00"/>
```

465 **6.1.6.12 Interval**

466 XML uses schema type xsd:unsignedInt

```
467 <Offset type="Interval" value="27"/>
```

468

469

470 7 XML Profile Test Cases

471 This section contains a test case that demonstrates the XML profile encoding using test case 12.1 from
472 [KMIP-TC] using protocol version 1.0 which exercises the Query operation and the Maximum Response
473 Size header field.

474 7.1.1 MSGENC-XML-1-10 - Query, Maximum Response Size

475 Perform a Query operation, querying the Operations and Objects supported by the server, with a
476 restriction on the Maximum Response Size set in the request header. Since the resulting Query response
477 is too big, an error is returned. Increase the Maximum Response Size, resubmit the Query request, and
478 get a successful response.

479 The specific list of operations and object types returned in the response MAY vary.

0001	<pre># TIME 0 <RequestMessage> <RequestHeader> <ProtocolVersion> <ProtocolVersionMajor type="Integer" value="1"/> <ProtocolVersionMinor type="Integer" value="0"/> </ProtocolVersion> <MaximumResponseSize type="Integer" value="256"/> <BatchCount type="Integer" value="1"/> </RequestHeader> <BatchItem> <Operation type="Enumeration" value="Query"/> <RequestPayload> <QueryFunction type="Enumeration" value="QueryOperations"/> <QueryFunction type="Enumeration" value="QueryObjects"/> </RequestPayload> </BatchItem> </RequestMessage></pre>
0018	<pre>42007801000000904200770100000048420069010000002042006a02000000040000000100000000 42006b02000000040000000000000000420050020000000400000100000000004200d0200000004 000000010000000042000f010000003842005c050000000400000018000000004200790100000020 4200740500000004000000010000000042007405000000040000000200000000</pre>
0019	<pre><ResponseMessage> <ResponseHeader> <ProtocolVersion> <ProtocolVersionMajor type="Integer" value="1"/> <ProtocolVersionMinor type="Integer" value="0"/> </ProtocolVersion> <TimeStamp type="DateTime" value="2013-06-26T09:09:17+00:00"/> <BatchCount type="Integer" value="1"/> </ResponseHeader> <BatchItem> <Operation type="Enumeration" value="Query"/> <ResultStatus type="Enumeration" value="OperationFailed"/> <ResultReason type="Enumeration" value="ResponseTooLarge"/> <ResultMessage type="TextString" value="TOO_LARGE"/> </BatchItem> </ResponseMessage></pre>
0032	<pre>42007b01000000a042007a0100000048420069010000002042006a02000000040000000100000000 42006b0200000004000000000000000042009209000000080000000051caafbd4200d0200000004 000000010000000042000f010000004842005c0500000004000000180000000042007f0500000004</pre>

	000000010000000042007e0500000004000000020000000042007d0700000009544f4f5f4c4152474500000000000000
<pre> 0032 0033 0034 0035 0036 0037 0038 0039 0040 0041 0042 0043 0044 0045 0046 0047 0048 </pre>	<pre> # TIME 1 <RequestMessage> <RequestHeader> <ProtocolVersion> <ProtocolVersionMajor type="Integer" value="1"/> <ProtocolVersionMinor type="Integer" value="0"/> </ProtocolVersion> <MaximumResponseSize type="Integer" value="2048"/> <BatchCount type="Integer" value="1"/> </RequestHeader> <BatchItem> <Operation type="Enumeration" value="Query"/> <RequestPayload> <QueryFunction type="Enumeration" value="QueryOperations"/> <QueryFunction type="Enumeration" value="QueryObjects"/> </RequestPayload> </BatchItem> </RequestMessage> 42007801000000904200770100000048420069010000002042006a02000000040000000100000000 42006b020000000400000000000000004200500200000004000008000000000042000d0200000004 000000010000000042000f010000003842005c050000000400000018000000004200790100000020 4200740500000004000000010000000042007405000000040000000200000000 </pre>
<pre> 0049 0050 0051 0052 0053 0054 0055 0056 0057 0058 0059 0060 0061 0062 0063 0064 0065 0066 0067 0068 0069 0070 0071 0072 0073 0074 0075 0076 0077 0078 </pre>	<pre> <ResponseMessage> <ResponseHeader> <ProtocolVersion> <ProtocolVersionMajor type="Integer" value="1"/> <ProtocolVersionMinor type="Integer" value="0"/> </ProtocolVersion> <TimeStamp type="DateTime" value="2013-06-26T09:09:17+00:00"/> <BatchCount type="Integer" value="1"/> </ResponseHeader> <BatchItem> <Operation type="Enumeration" value="Query"/> <ResponseStatus type="Enumeration" value="Success"/> <ResponsePayload> <Operation type="Enumeration" value="Query"/> <Operation type="Enumeration" value="Locate"/> <Operation type="Enumeration" value="Destroy"/> <Operation type="Enumeration" value="Get"/> <Operation type="Enumeration" value="Create"/> <Operation type="Enumeration" value="Register"/> <Operation type="Enumeration" value="GetAttributes"/> <Operation type="Enumeration" value="GetAttributeList"/> <Operation type="Enumeration" value="AddAttribute"/> <Operation type="Enumeration" value="ModifyAttribute"/> <Operation type="Enumeration" value="DeleteAttribute"/> <Operation type="Enumeration" value="Activate"/> <Operation type="Enumeration" value="Revoke"/> <Operation type="Enumeration" value="Poll"/> <Operation type="Enumeration" value="Cancel"/> <Operation type="Enumeration" value="Check"/> <Operation type="Enumeration" value="GetUsageAllocation"/> </ResponsePayload> </BatchItem> </ResponseMessage> </pre>

```
0079 <Operation type="Enumeration" value="CreateKeyPair"/>
0080 <Operation type="Enumeration" value="ReKey"/>
0081 <Operation type="Enumeration" value="Archive"/>
0082 <Operation type="Enumeration" value="Recover"/>
0083 <Operation type="Enumeration" value="ObtainLease"/>
0084 <Operation type="Enumeration" value="Certify"/>
0085 <Operation type="Enumeration" value="ReCertify"/>
0086 <Operation type="Enumeration" value="Notify"/>
0087 <Operation type="Enumeration" value="Put"/>
0088 <ObjectType type="Enumeration" value="Certificate"/>
0089 <ObjectType type="Enumeration" value="SymmetricKey"/>
0090 <ObjectType type="Enumeration" value="SecretData"/>
0091 <ObjectType type="Enumeration" value="PublicKey"/>
0092 <ObjectType type="Enumeration" value="PrivateKey"/>
0093 <ObjectType type="Enumeration" value="Template"/>
0094 <ObjectType type="Enumeration" value="OpaqueObject"/>
0095 <ObjectType type="Enumeration" value="SplitKey"/>
0096 </ResponsePayload>
0097 </BatchItem>
0098 </ResponseMessage>

42007b01000002a042007a0100000048420069010000002042006a02000000040000000100000000
42006b020000000400000000000000042009209000000080000000051caafbd42000d0200000004
000000010000000042000f010000024842005c0500000004000000180000000042007f0500000004
000000000000000042007c010000022042005c0500000004000000180000000042005c0500000004
000000080000000042005c0500000004000000140000000042005c05000000040000000a00000000
42005c0500000004000000010000000042005c0500000004000000030000000042005c0500000004
0000000b0000000042005c05000000040000000c0000000042005c05000000040000000d00000000
42005c05000000040000000e0000000042005c05000000040000000f0000000042005c0500000004
000000120000000042005c0500000004000000130000000042005c05000000040000001a00000000
42005c0500000004000000190000000042005c0500000004000000090000000042005c0500000004
000000110000000042005c0500000004000000020000000042005c05000000040000000400000000
42005c0500000004000000150000000042005c0500000004000000160000000042005c0500000004
000000100000000042005c0500000004000000060000000042005c05000000040000000700000000
42005c05000000040000001b0000000042005c05000000040000001c00000000420057050000004
00000001000000004200570500000004000000020000000042005705000000040000000700000000
42005705000000040000000300000000420057050000000400000004000000004200570500000004
00000006000000004200570500000004000000080000000042005705000000040000000500000000
```

481 8 Conformance

482 8.1 HTTPS Profile Conformance

483 KMIP client and server implementations conformant to this profile:

- 484 1. SHALL support the Authentication Suite conditions as specified in Section 2.1 of this profile.
- 485 2. SHALL support the KMIP Port Number conditions as specified in Section 2.2 of this profile.
- 486 3. SHALL support the Request URL conditions as specified in Section 2.3 of this profile.
- 487 4. SHALL support the HTTP Encoding conditions as specified in Section 2.4 of this profile.
- 488 5. SHALL support mapping of all TTLV tags and enumerations specified within each version of the
489 [KMIP-SPEC] that is supported.
- 490 6. SHALL support user defined extensions containing additional tags and enumerations not
491 specified within [KMIP-SPEC].

492 8.2 JSON Profile Conformance

493 KMIP client and server implementations conformant to this profile:

- 494 1. SHALL support JSON message encoding for request and response messages as specified in
495 Section 4.1 of this profile.
- 496 2. SHALL support mapping of all TTLV tags and enumerations specified within each version of the
497 [KMIP-SPEC] that is supported
- 498 3. SHALL support user defined extensions containing additional tags and enumerations not
499 specified within [KMIP-SPEC]

500 8.3 XML Profile Conformance

501 KMIP client and server implementations conformant to this profile:

- 502 1. SHALL support XML message encoding for request and response messages as specified in
503 Section 6.1 of this profile.
- 504 2. SHALL support mapping of all TTLV tags and enumerations specified within each version of the
505 [KMIP-SPEC] that is supported.
- 506 3. SHALL support user defined extensions containing additional tags and enumerations not
507 specified within [KMIP-SPEC].

508 8.4 Permitted Test Case Variations

509 Whilst the test cases provided in this Profile define the allowed request and response content, some
510 inherent variations MAY occur and are permitted within a successfully completed test case.

511 Each test case MAY include allowed variations in the description of the test case in addition to the
512 variations noted in this section.

513 Other variations not explicitly noted in this Profile SHALL be deemed non-conformant.

514 8.4.1 Variable Items

515 An implementation conformant to this Profile MAY vary the following values:

- 516 1. UniqueIdentifier
- 517 2. PrivateKeyUniqueIdentifier
- 518 3. PublicKeyUniqueIdentifier

- 519 4. UniqueBatchItemIdentifier
520 5. AsynchronousCorrelationValue
521 6. TimeStamp
522 7. KeyValue / KeyMaterial including:
523 a. key material content returned for managed cryptographic objects which are generated by
524 the server
525 b. wrapped versions of keys where the wrapping key is dynamic or the wrapping contains
526 variable output for each wrap operation
527 8. For response containing the output of cryptographic operation in Data / SignatureData/ MACData
528 / IVCounterNonce where:
529 a. the managed object is generated by the server; or
530 b. the operation inherently contains variable output
531 9. For the following DateTime attributes where the value is not specified in the request as a fixed
532 DateTime value:
533 a. ActivationDate
534 b. ArchiveDate
535 c. CompromiseDate
536 d. CompromiseOccurrenceDate
537 e. DeactivationDate
538 f. DestroyDate
539 g. InitialDate
540 h. LastChangeDate
541 i. ProtectStartDate
542 j. ProcessStopDate
543 k. ValidityDate
544 l. OriginalCreationDate
545 10. LinkedObjectIdentifier
546 11. DigestValue
547 a. For those managed cryptographic objects which are dynamically generated
548 12. KeyFormatType
549 a. The key format type selected by the server when it creates managed objects
550 13. Digest
551 a. The HashingAlgorithm selected by the server when it calculates the digest for a managed
552 object for which it has access to the key material
553 b. The Digest Value
554 14. Extensions reported in Query for ExtensionList and ExtensionMap
555 15. Application Namespaces reported in Query
556 16. Object Types reported in Query other than those noted as required in this profile
557 17. Operation Types reported in Query other than those noted as required in this profile (or any
558 referenced profile documents)
559 18. For TextString attribute values containing test identifiers:
560 a. Additional vendor or application prefixes
561 19. Additional attributes beyond those noted in the response
562

- 563 An implementation conformant to this Profile MAY allow the following response variations:
- 564 1. Object Group values – May or may not return one or more Object Group values not included in
565 the requests
 - 566 2. y-CustomAttributes – May or may not include additional server-specific associated attributes not
567 included in requests
 - 568 3. Message Extensions – May or may not include additional (non-critical) vendor extensions
 - 569 4. TemplateAttribute – May or may not be included in responses where the Template Attribute
570 response is noted as optional in [KMIP-SPEC]
 - 571 5. AttributeIndex – May or may not include Attribute Index value where the Attribute Index value is 0
572 for Protocol Versions 1.1 and above.
 - 573 6. ResultMessage – May or may not be included in responses and the value (if included) may vary
574 from the text contained within the test case.
 - 575 7. The list of Protocol Versions returned in a DiscoverVersion response may include additional
576 protocol versions if the request has not specified a list of client supported Protocol Versions.
 - 577 8. VendorIdentification - The value (if included) may vary from the text contained within the test
578 case.

579 **8.4.2 Variable behavior**

- 580 An implementation conformant to this Profile SHALL allow variation of the following behavior:
- 581 1. A test may omit the clean-up requests and responses (containing Revoke and/or Destroy) at the
582 end of the test provided there is a separate mechanism to remove the created objects during
583 testing.
 - 584 2. A test may omit the test identifiers if the client is unable to include them in requests. This includes
585 the following attributes:
 - 586 a. Name; and
 - 587 b. x-ID
- 588
- 589

Appendix A. Acknowledgments

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

591 **Original HTTPS Profile Proposal:**

592 Alan Frindell, SafeNet, Inc.

593

594 **Original HTTPS Profile Contributors:**

595 Mathias Björkqvist, IBM

Participants:

596 Hal Aldridge, Sypris Electronics
597 Mike Allen, Symantec
598 Gordon Arnold, IBM
599 Todd Arnold, IBM
600 Richard Austin, Hewlett-Packard
601 Lars Bagnert, PrimeKey
602 Elaine Barker, NIST
603 Peter Bartok, Venafi, Inc.
604 Tom Benjamin, IBM
605 Anthony Berglas, Cryptsoft
606 Mathias Björkqvist, IBM
607 Kevin Bocket, Venafi
608 Anne Bolgert, IBM
609 Alan Brown, Thales e-Security
610 Tim Bruce, CA Technologies
611 Chris Burchett, Credant Technologies, Inc.
612 Kelley Burgin, National Security Agency
613 Robert Burns, Thales e-Security
614 Chuck Castleton, Venafi
615 Kenli Chong, QuintessenceLabs
616 John Clark, Hewlett-Packard
617 Tom Clifford, Symantec Corp.
618 Doron Cohen, SafeNet, Inc
619 Tony Cox, Cryptsoft
620 Russell Dietz, SafeNet, Inc
621 Graydon Dodson, Lexmark International Inc.
622 Vinod Duggirala, EMC Corporation
623 Chris Dunn, SafeNet, Inc.
624 Michael Duren, Sypris Electronics
625 James Dzierzanowski, American Express CCoE
626 Faisal Faruqui, Thales e-Security
627 Stan Feather, Hewlett-Packard
628 David Finkelstein, Symantec Corp.
629 James Fitzgerald, SafeNet, Inc.
630 Indra Fitzgerald, Hewlett-Packard
631 Judith Furlong, EMC Corporation
632 Susan Gleeson, Oracle
633 Robert Griffin, EMC Corporation
634 Paul Grojean, Individual
635 Robert Haas, IBM
636 Thomas Hardjono, M.I.T.
637 ChengDong He, Huawei Technologies Co., Ltd.
638 Steve He, Vormetric

639 Kurt Heberlein, Hewlett-Packard
640 Larry Hofer, Emulex Corporation
641 Maryann Hondo, IBM
642 Walt Hubis, NetApp
643 Tim Hudson, Cryptsoft
644 Jonas Iggbom, Venafi, Inc.
645 Sitaram Inguva, American Express CCoE
646 Jay Jacobs, Target Corporation
647 Glen Jaquette, IBM
648 Mahadev Karadiguddi, NetApp
649 Greg Kazmierczak, Wave Systems Corp.
650 Marc Kenig, SafeNet, Inc.
651 Mark Knight, Thales e-Security
652 Kathy Kriese, Symantec Corporation
653 Mark Lambiase, SecureAuth
654 John Leiseboer, Quintessence Labs
655 Hal Lockhart, Oracle Corporation
656 Robert Lockhart, Thales e-Security
657 Anne Luk, Cryptsoft
658 Sairam Manidi, Freescale
659 Luther Martin, Voltage Security
660 Neil McEvoy, iFOSSF
661 Marina Milshtein, Individual
662 Dale Moberg, Axway Software
663 Jishnu Mukeri, Hewlett-Packard
664 Bryan Olson, Hewlett-Packard
665 John Peck, IBM
666 Rob Philpott, EMC Corporation
667 Denis Pochuev, SafeNet, Inc.
668 Reid Poole, Venafi, Inc.
669 Ajai Puri, SafeNet, Inc.
670 Saravanan Ramalingam, Thales e-Security
671 Peter Reed, SafeNet, Inc.
672 Bruce Rich, IBM
673 Christina Richards, American Express CCoE
674 Warren Robbins, Dell
675 Peter Robinson, EMC Corporation
676 Scott Rotondo, Oracle
677 Saikat Saha, SafeNet, Inc.
678 Anil Saldhana, Red Hat
679 Subhash Sankuratipati, NetApp
680 Boris Schumperli, Cryptomathic
681 Greg Singh, QuintessenceLabs
682 David Smith, Venafi, Inc
683 Brian Spector, Certivox
684 Terence Spies, Voltage Security
685 Deborah Steckroth, RouteOne LLC
686 Michael Stevens, QuintessenceLabs
687 Marcus Streets, Thales e-Security
688 Satish Sundar, IBM
689 Kiran Thota, VMware
690 Somanchi Trinath, Freescale Semiconductor, Inc.
691 Nathan Turajski, Thales e-Security
692 Sean Turner, IECA, Inc.
693 Paul Turner, Venafi, Inc.
694 Rod Wideman, Quantum Corporation
695 Steven Wierenga, Hewlett-Packard

696 Jin Wong, QuintessenceLabs
697 Sameer Yami, Thales e-Security
698 Peter Yee, EMC Corporation
699 Krishna Yellepeddy, IBM
700 Catherine Ying, SafeNet, Inc.
701 Tatu Ylonen, SSH Communications Security (Tectia Corp)
702 Michael Yoder, Vormetric. Inc.
703 Magda Zdunkiewicz, Cryptsoft
704 Peter Zelechowski, Election Systems & Software

Appendix B. KMIP Specification Cross Reference

Reference Term	KMIP 1.0	KMIP 1.1	KMIP 1.2
1 Introduction			
<i>Non-Normative References</i>	1.3.	1.3.	1.3.
<i>Normative References</i>	1.2.	1.2.	1.2.
<i>Terminology</i>	1.1.	1.1.	1.1.
2 Objects			
<i>Attribute</i>	2.1.1.	2.1.1.	2.1.1.
<i>Base Objects</i>	2.1.	2.1.	2.1.
<i>Certificate</i>	2.2.1.	2.2.1.	2.2.1.
<i>Credential</i>	2.1.2.	2.1.2.	2.1.2.
<i>Data</i>	-	-	2.1.10.
<i>Data Length</i>	-	-	2.1.11.
<i>Extension Information</i>	-	2.1.9.	2.1.9.
<i>Key Block</i>	2.1.3.	2.1.3.	2.1.3.
<i>Key Value</i>	2.1.4.	2.1.4.	2.1.4.
<i>Key Wrapping Data</i>	2.1.5.	2.1.5.	2.1.5.
<i>Key Wrapping Specification</i>	2.1.6.	2.1.6.	2.1.6.
<i>MAC Data</i>	-	-	2.1.13.
<i>Managed Objects</i>	2.2.	2.2.	2.2.
<i>Nonce</i>	-	-	2.1.14.
<i>Opaque Object</i>	2.2.8.	2.2.8.	2.2.8.
<i>PGP Key</i>	-	-	2.2.9.
<i>Private Key</i>	2.2.4.	2.2.4.	2.2.4.
<i>Public Key</i>	2.2.3.	2.2.3.	2.2.3.
<i>Secret Data</i>	2.2.7.	2.2.7.	2.2.7.
<i>Signature Data</i>	-	-	2.1.12.
<i>Split Key</i>	2.2.5.	2.2.5.	2.2.5.
<i>Symmetric Key</i>	2.2.2.	2.2.2.	2.2.2.
<i>Template</i>	2.2.6.	2.2.6.	2.2.6.
<i>Template-Attribute Structures</i>	2.1.8.	2.1.8.	2.1.8.
<i>Transparent DH Private Key</i>	2.1.7.6.	2.1.7.6.	2.1.7.6.
<i>Transparent DH Public Key</i>	2.1.7.7.	2.1.7.7.	2.1.7.7.
<i>Transparent DSA Private Key</i>	2.1.7.2.	2.1.7.2.	2.1.7.2.
<i>Transparent DSA Public Key</i>	2.1.7.3.	2.1.7.3.	2.1.7.3.
<i>Transparent ECDH Private Key</i>	2.1.7.10.	2.1.7.10.	2.1.7.10.
<i>Transparent ECDH Public Key</i>	2.1.7.11.	2.1.7.11.	2.1.7.11.
<i>Transparent ECDSA Private Key</i>	2.1.7.8.	2.1.7.8.	2.1.7.8.
<i>Transparent ECDSA Public Key</i>	2.1.7.9.	2.1.7.9.	2.1.7.9.
<i>Transparent ECMQV Private Key</i>	2.1.7.12.	2.1.7.12.	2.1.7.12.
<i>Transparent ECMQV Public Key</i>	2.1.7.13.	2.1.7.13.	2.1.7.13.
<i>Transparent Key Structures</i>	2.1.7.	2.1.7.	2.1.7.
<i>Transparent RSA Private Key</i>	2.1.7.4.	2.1.7.4.	2.1.7.4.
<i>Transparent RSA Public Key</i>	2.1.7.5.	2.1.7.5.	2.1.7.5.
<i>Transparent Symmetric Key</i>	2.1.7.1.	2.1.7.1.	2.1.7.1.
3 Attributes			
<i>Activation Date</i>	3.19.	3.24.	3.24.
<i>Alternative Name</i>	-	-	3.40.
<i>Application Specific Information</i>	3.30.	3.36.	3.36.
<i>Archive Date</i>	3.27.	3.32.	3.32.

Reference Term	KMIP 1.0	KMIP 1.1	KMIP 1.2
<i>Attributes</i>	3	3	3
<i>Certificate Identifier</i>	3.9.	3.13.	3.13.
<i>Certificate Issuer</i>	3.11.	3.15.	3.15.
<i>Certificate Length</i>	-	3.9.	3.9.
<i>Certificate Subject</i>	3.10.	3.14.	3.14.
<i>Certificate Type</i>	3.8.	3.8.	3.8.
<i>Compromise Date</i>	3.25.	3.30.	3.30.
<i>Compromise Occurrence Date</i>	3.24.	3.29.	3.29.
<i>Contact Information</i>	3.31.	3.37.	3.37.
<i>Cryptographic Algorithm</i>	3.4.	3.4.	3.4.
<i>Cryptographic Domain Parameters</i>	3.7.	3.7.	3.7.
<i>Cryptographic Length</i>	3.5.	3.5.	3.5.
<i>Cryptographic Parameters</i>	3.6.	3.6.	3.6.
<i>Custom Attribute</i>	3.33.	3.39.	3.39.
<i>Deactivation Date</i>	3.22.	3.27.	3.27.
<i>Default Operation Policy</i>	3.13.2.	3.18.2.	3.18.2.
<i>Default Operation Policy for Certificates and Public Key Objects</i>	3.13.2.2.	3.18.2.2.	3.18.2.2.
<i>Default Operation Policy for Secret Objects</i>	3.13.2.1.	3.18.2.1.	3.18.2.1.
<i>Default Operation Policy for Template Objects</i>	3.13.2.3.	3.18.2.3.	3.18.2.3.
<i>Destroy Date</i>	3.23.	3.28.	3.28.
<i>Digest</i>	3.12.	3.17.	3.17.
<i>Digital Signature Algorithm</i>	-	3.16.	3.16.
<i>Fresh</i>	-	3.34.	3.34.
<i>Initial Date</i>	3.18.	3.23.	3.23.
<i>Key Value Location</i>	-	-	3.42.
<i>Key Value Present</i>	-	-	3.41.
<i>Last Change Date</i>	3.32.	3.38.	3.38.
<i>Lease Time</i>	3.15.	3.20.	3.20.
<i>Link</i>	3.29.	3.35.	3.35.
<i>Name</i>	3.2.	3.2.	3.2.
<i>Object Group</i>	3.28.	3.33.	3.33.
<i>Object Type</i>	3.3.	3.3.	3.3.
<i>Operation Policy Name</i>	3.13.	3.18.	3.18.
<i>Operations outside of operation policy control</i>	3.13.1.	3.18.1.	3.18.1.
<i>Original Creation Date</i>	-	-	3.43.
<i>Process Start Date</i>	3.20.	3.25.	3.25.
<i>Protect Stop Date</i>	3.21.	3.26.	3.26.
<i>Revocation Reason</i>	3.26.	3.31.	3.31.
<i>State</i>	3.17.	3.22.	3.22.
<i>Unique Identifier</i>	3.1.	3.1.	3.1.
<i>Usage Limits</i>	3.16.	3.21.	3.21.
<i>X.509 Certificate Identifier</i>	-	3.10.	3.10.
<i>X.509 Certificate Issuer</i>	-	3.12.	3.12.
<i>X.509 Certificate Subject</i>	-	3.11.	3.11.
4 Client-to-Server Operations			
<i>Activate</i>	4.18.	4.19.	4.19.
<i>Add Attribute</i>	4.13.	4.14.	4.14.
<i>Archive</i>	4.21.	4.22.	4.22.
<i>Cancel</i>	4.25.	4.27.	4.27.
<i>Certify</i>	4.6.	4.7.	4.7.
<i>Check</i>	4.9.	4.10.	4.10.
<i>Create</i>	4.1.	4.1.	4.1.
<i>Create Key Pair</i>	4.2.	4.2.	4.2.

Reference Term	KMIP 1.0	KMIP 1.1	KMIP 1.2
<i>Create Split Key</i>	-	-	4.38.
<i>Decrypt</i>	-	-	4.30.
<i>Delete Attribute</i>	4.15.	4.16.	4.16.
<i>Derive Key</i>	4.5.	4.6.	4.6.
<i>Destroy</i>	4.20.	4.21.	4.21.
<i>Discover Versions</i>	-	4.26.	4.26.
<i>Encrypt</i>	-	-	4.29.
<i>Get</i>	4.10.	4.11.	4.11.
<i>Get Attribute List</i>	4.12.	4.13.	4.13.
<i>Get Attributes</i>	4.11.	4.12.	4.12.
<i>Get Usage Allocation</i>	4.17.	4.18.	4.18.
<i>Hash</i>	-	-	4.37.
<i>Join Split Key</i>	-	-	4.39.
<i>Locate</i>	4.8.	4.9.	4.9.
<i>MAC</i>	-	-	4.33.
<i>MAC Verify</i>	-	-	4.34.
<i>Modify Attribute</i>	4.14.	4.15.	4.15.
<i>Obtain Lease</i>	4.16.	4.17.	4.17.
<i>Poll</i>	4.26.	4.28.	4.28.
<i>Query</i>	4.24.	4.25.	4.25.
<i>Re-certify</i>	4.7.	4.8.	4.8.
<i>Recover</i>	4.22.	4.23.	4.23.
<i>Register</i>	4.3.	4.3.	4.3.
<i>Re-key</i>	4.4.	4.4.	4.4.
<i>Re-key Key Pair</i>	-	4.5.	4.5.
<i>Revoke</i>	4.19.	4.20.	4.20.
<i>RNG Retrieve</i>	-	-	4.35.
<i>RNG Seed</i>	-	-	4.36.
<i>Sign</i>	-	-	4.31.
<i>Signature Verify</i>	-	-	4.32.
<i>Validate</i>	4.23.	4.24.	4.24.
5 Server-to-Client Operations			
<i>Notify</i>	5.1.	5.1.	5.1.
<i>Put</i>	5.2.	5.2.	5.2.
6 Message Contents			
<i>Asynchronous Correlation Value</i>	6.8.	6.8.	6.8.
<i>Asynchronous Indicator</i>	6.7.	6.7.	6.7.
<i>Attestation Capable Indicator</i>	-	-	6.17.
<i>Batch Count</i>	6.14.	6.14.	6.14.
<i>Batch Error Continuation Option</i>	6.13.	6.13.	6.13.
<i>Batch Item</i>	6.15.	6.15.	6.15.
<i>Batch Order Option</i>	6.12.	6.12.	6.12.
<i>Maximum Response Size</i>	6.3.	6.3.	6.3.
<i>Message Extension</i>	6.16.	6.16.	6.16.
<i>Operation</i>	6.2.	6.2.	6.2.
<i>Protocol Version</i>	6.1.	6.1.	6.1.
<i>Result Message</i>	6.11.	6.11.	6.11.
<i>Result Reason</i>	6.10.	6.10.	6.10.
<i>Result Status</i>	6.9.	6.9.	6.9.
<i>Time Stamp</i>	6.5.	6.5.	6.5.
<i>Unique Batch Item ID</i>	6.4.	6.4.	6.4.
7 Message Format			

Reference Term	KMIP 1.0	KMIP 1.1	KMIP 1.2
<i>Message Structure</i>	7.1.	7.1.	7.1.
<i>Operations</i>	7.2.	7.2.	7.2.
8 Authentication			
<i>Authentication</i>	8	8	8
9 Message Encoding			
<i>Alternative Name Type Enumeration</i>	-	-	9.1.3.2.34.
<i>Attestation Type Enumeration</i>	-	-	9.1.3.2.36.
<i>Batch Error Continuation Option Enumeration</i>	9.1.3.2.29.	9.1.3.2.30.	9.1.3.2.30.
<i>Bit Masks</i>	9.1.3.3.	9.1.3.3.	9.1.3.3.
<i>Block Cipher Mode Enumeration</i>	9.1.3.2.13.	9.1.3.2.14.	9.1.3.2.14.
<i>Cancellation Result Enumeration</i>	9.1.3.2.24.	9.1.3.2.25.	9.1.3.2.25.
<i>Certificate Request Type Enumeration</i>	9.1.3.2.21.	9.1.3.2.22.	9.1.3.2.22.
<i>Certificate Type Enumeration</i>	9.1.3.2.6.	9.1.3.2.6.	9.1.3.2.6.
<i>Credential Type Enumeration</i>	9.1.3.2.1.	9.1.3.2.1.	9.1.3.2.1.
<i>Cryptographic Algorithm Enumeration</i>	9.1.3.2.12.	9.1.3.2.13.	9.1.3.2.13.
<i>Cryptographic Usage Mask</i>	9.1.3.3.1.	9.1.3.3.1.	9.1.3.3.1.
<i>Defined Values</i>	9.1.3.	9.1.3.	9.1.3.
<i>Derivation Method Enumeration</i>	9.1.3.2.20.	9.1.3.2.21.	9.1.3.2.21.
<i>Digital Signature Algorithm Enumeration</i>	-	9.1.3.2.7.	9.1.3.2.7.
<i>Encoding Option Enumeration</i>	-	9.1.3.2.32.	9.1.3.2.32.
<i>Enumerations</i>	9.1.3.2.	9.1.3.2.	9.1.3.2.
<i>Examples</i>	9.1.2.	9.1.2.	9.1.2.
<i>Hashing Algorithm Enumeration</i>	9.1.3.2.15.	9.1.3.2.16.	9.1.3.2.16.
<i>Item Length</i>	9.1.1.3.	9.1.1.3.	9.1.1.3.
<i>Item Tag</i>	9.1.1.1.	9.1.1.1.	9.1.1.1.
<i>Item Type</i>	9.1.1.2.	9.1.1.2.	9.1.1.2.
<i>Item Value</i>	9.1.1.4.	9.1.1.4.	9.1.1.4.
<i>Key Compression Type Enumeration</i>	9.1.3.2.2.	9.1.3.2.2.	9.1.3.2.2.
<i>Key Format Type Enumeration</i>	9.1.3.2.3.	9.1.3.2.3.	9.1.3.2.3.
<i>Key Role Type Enumeration</i>	9.1.3.2.16.	9.1.3.2.17.	9.1.3.2.17.
<i>Key Value Location Type Enumeration</i>	-	-	9.1.3.2.35.
<i>Link Type Enumeration</i>	9.1.3.2.19.	9.1.3.2.20.	9.1.3.2.20.
<i>Name Type Enumeration</i>	9.1.3.2.10.	9.1.3.2.11.	9.1.3.2.11.
<i>Object Group Member Enumeration</i>	-	9.1.3.2.33.	9.1.3.2.33.
<i>Object Type Enumeration</i>	9.1.3.2.11.	9.1.3.2.12.	9.1.3.2.12.
<i>Opaque Data Type Enumeration</i>	9.1.3.2.9.	9.1.3.2.10.	9.1.3.2.10.
<i>Operation Enumeration</i>	9.1.3.2.26.	9.1.3.2.27.	9.1.3.2.27.
<i>Padding Method Enumeration</i>	9.1.3.2.14.	9.1.3.2.15.	9.1.3.2.15.
<i>Put Function Enumeration</i>	9.1.3.2.25.	9.1.3.2.26.	9.1.3.2.26.
<i>Query Function Enumeration</i>	9.1.3.2.23.	9.1.3.2.24.	9.1.3.2.24.
<i>Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV</i>	9.1.3.2.5.	9.1.3.2.5.	9.1.3.2.5.
<i>Result Reason Enumeration</i>	9.1.3.2.28.	9.1.3.2.29.	9.1.3.2.29.
<i>Result Status Enumeration</i>	9.1.3.2.27.	9.1.3.2.28.	9.1.3.2.28.
<i>Revocation Reason Code Enumeration</i>	9.1.3.2.18.	9.1.3.2.19.	9.1.3.2.19.
<i>Secret Data Type Enumeration</i>	9.1.3.2.8.	9.1.3.2.9.	9.1.3.2.9.
<i>Split Key Method Enumeration</i>	9.1.3.2.7.	9.1.3.2.8.	9.1.3.2.8.
<i>State Enumeration</i>	9.1.3.2.17.	9.1.3.2.18.	9.1.3.2.18.
<i>Storage Status Mask</i>	9.1.3.3.2.	9.1.3.3.2.	9.1.3.3.2.
<i>Tags</i>	9.1.3.1.	9.1.3.1.	9.1.3.1.
<i>TTLV Encoding</i>	9.1.	9.1.	9.1.
<i>TTLV Encoding Fields</i>	9.1.1.	9.1.1.	9.1.1.
<i>Usage Limits Unit Enumeration</i>	9.1.3.2.30.	9.1.3.2.31.	9.1.3.2.31.

Reference Term	KMIP 1.0	KMIP 1.1	KMIP 1.2
<i>Validity Indicator Enumeration</i>	9.1.3.2.22.	9.1.3.2.23.	9.1.3.2.23.
<i>Wrapping Method Enumeration</i>	9.1.3.2.4.	9.1.3.2.4.	9.1.3.2.4.
<i>XML Encoding</i>	9.2.	-	-
10 Transport			
<i>Transport</i>	10	10	10
12 KMIP Server and Client Implementation Conformance			
<i>Conformance clauses for a KMIP Server</i>	12.1.	-	-
<i>KMIP Client Implementation Conformance</i>	-	12.2.	12.2.
<i>KMIP Server Implementation Conformance</i>	-	12.1.	12.1.

705 **Appendix C. Revision History**

706

Revision	Date	Editor	Changes Made
wd01	26-June-2013	Tim Hudson	Merged version of the three committee draft documents. Updated conformance wording style. Updated test case style. Applied new OASIS template.
wd02	6-August-2013	Tim Hudson	Updated to include Permitted Test Case Variations
wd03	10-August-2013	Tim Hudson	Updated Permitted Test Case Variations

707