KMIP Additional Message Encodings Version 1.0

Committee Specification Draft 01

09 January 2014

Specification URIs

This version:

http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csd01/kmip-addtl-msg-enc-v1.0-csd01.doc (Authoritative)
http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csd01/kmip-addtl-msg-enc-v1.0-csd01.html
http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csd01/kmip-addtl-msg-enc-v1.0-csd01.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. http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/csd01/kmip-addtl-msg-enc-v1.0-csd01.html. Latest version: http://docs.oasis-open.org/kmip/kmip-addtl-msg-enc/v1.0/kmip-addtl-msg-enc-v1.0.html.

Notices

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

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:// 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 199 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.

[KMIP-PROF-1_1] *Key Management Interoperability Protocol Usage Guide Version 1.1*.
http://docs.oasis-open.org/kmip/profiles/v1.1/os/kmip-profiles-v1.1-os.doc
OASIS Standard 01. 24 January 2013.

[KMIP-PROF-1_2] *Key Management Interoperability Protocol Usage Guide Version 1.2*.
URL
Candidate OASIS Standard 01. DD MMM YYYY.

1.3 Non-Normative References

[KMIP-UG-1_0] *Key Management Interoperability Protocol Usage Guide Version 1.0.*
http://docs.oasis-open.org/kmip/ug/v1.1/kmip-ug-v1.1-cnd01.doc
Committee Note Draft, 1 December 2011

[KMIP-UG-1_1] *Key Management Interoperability Protocol Usage Guide Version 1.1.*
http://docs.oasis-open.org/kmip/ug/v1.1/cn01/kmip-ug-v1.1-cn01.doc
Committee Note 01, 27 July 2012

[KMIP-UG-1_2] *Key Management Interoperability Protocol Usage Guide Version 1.2.*
URL
Committee Note Draft, DD MMM YYYY

[KMIP-TC-1_1] *Key Management Interoperability Protocol Test Cases Version 1.1.*
http://docs.oasis-open.org/kmip/testcases/v1.1/cn01/kmip-testcases-v1.1-cn01.doc, Committee Note 01, 27 July 2012.

[KMIP-TC-1_2] *Key Management Interoperability Protocol Test Cases Version 1.2.*
URL, Committee Note Draft, DD MMM YYYY.

[KMIP-UC] *Key Management Interoperability Protocol Use Cases Version 1.0.*
http://docs.oasis-open.org/kmip/usecases/v1.0/cs01/kmip-usecases-1.0-cs-01.doc, Committee Specification, 15 June 2010.

2 HTTPS Profile

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

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

2.1 Authentication Suite

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

2.2 KMIP Port Number

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

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

2.3 Request URI

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

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

2.4 HTTP Encoding

KMIP client implementations conformant to this profile:

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

KMIP server implementations conformant to this profile:

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

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

3 HTTPS Profile Test Cases

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

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

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

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>

      420078010000009042007701000000484200690100000020420 06a020000000400000001000000000
      42006b020000000400000000000000004200500200000004000001000000000042000d0200000004
      000000010000000042000f0100000038420 05c0500000004000000180000000042007901000000020
      420074050000000400000001000000004200740500000004000000020000000 0

      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 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>

        42007b01000000a042007a0100000048420069010000002042006a020000000400000001000000000
        42006b020000000400000000000000000420092090000000800000000051caafbd42000d0200000004
        000000010000000042000f0100000048420005c0500000004000000180000000042007f0500000004
        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 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 00 48   ........B......H
        000000b0: 42 00 5c 05 00 00 00 04-00 00 00 18 00 00 00 00   B.\.............
        000000c0: 42 00 7f 05 00 00 00 04-00 00 00 01 00 00 00 00   B...............
        000000d0: 42 00 7e 05 00 00 00 04-00 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.......
```

```
        # 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>

        420078010000009042007701000000484200690100000020420060200000004000000010000000
        42006b020000000400000000000000000420050020000000400000800000000000042000d0200000004
        000000010000000042000f0100000038420005c0500000004000000180000000042007901000000020
        420074050000000400000001000000004200740500000004000000020000000

        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 08 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 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-                          ........
```

```
<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="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"/>
      <Operation type="Enumeration" value="CreateKeyPair"/>
      <Operation type="Enumeration" value="ReKey"/>
      <Operation type="Enumeration" value="Archive"/>
      <Operation type="Enumeration" value="Recover"/>
      <Operation type="Enumeration" value="ObtainLease"/>
      <Operation type="Enumeration" value="Certify"/>
      <Operation type="Enumeration" value="ReCertify"/>
      <Operation type="Enumeration" value="Notify"/>
      <Operation type="Enumeration" value="Put"/>
      <ObjectType type="Enumeration" value="Certificate"/>
      <ObjectType type="Enumeration" value="SymmetricKey"/>
      <ObjectType type="Enumeration" value="SecretData"/>
      <ObjectType type="Enumeration" value="PublicKey"/>
      <ObjectType type="Enumeration" value="PrivateKey"/>
      <ObjectType type="Enumeration" value="Template"/>
```

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

42007b01000002a042007a0100000048420069010000002042006a020000000400000001000000000
42006b020000000400000000000000004200920900000008000000051caafbd42000d0200000004
000000010000000042000f01000002484200 5c0500000040000001800000000420 07f0500000004
0000000000000000420 07c01000002204200 5c0500000040000001800000000420 05c0500000004
0000000800000000420 05c0500000004000000140000000042005c0500000004000000a00000000
42005c0500000004000000010000000042005c0500000004000000030000000042005c0500000004
0000000b0000000042005c0500000004000000c0000000042005c0500000004000000d00000000
42005c0500000004000000e0000000042005c0500000004000000f0000000042005c0500000004
000000120000000042005c0500000004000000130000000042005c0500000004000001a00000000
42005c0500000004000000190000000042005c0500000004000000090000000042005c0500000004
000000110000000042005c0500000004000000020000000042005c0500000004000000400000000
42005c0500000004000000150000000042005c0500000004000000160000000042005c0500000004
000000100000000042005c0500000004000000060000000042005c0500000004000000070000000
42005c0500000004000001b0000000042005c0500000004000001c000000004200570500000004
000000010000000420057050000000400000002000000042005705000000040000000700000000
420057050000000400000003000000004200570500000004000000040000000042005705000000 04
000000060000000042005705000000040000000800000000420057050000000400000005000000 00

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 00 18 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

4 JSON Profile

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

4.1 JSON Encoding

4.1.1 Hex representations

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

4.1.2 Tags

Tags are a String that may contain either:

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

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

4.1.3 Normalizing Names

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

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

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

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

Example python name normalization code

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

```

Example perl name normalization code

4.1.4 Type

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

- Structure
- Integer
- LongInteger
- BigInteger
- Enumeration
- Boolean
- TextString
- ByteString
- DateTime
- Interval

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

4.1.5 Value

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

4.1.6 JSON Object

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

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

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

4.1.6.1 Tags

Tags are a String that may contain either:

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

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

```
{"tag": "0x420001", "type":"DateTime", "value":"2001-01-01T10:00:00+10:00"}
{"tag": "ActivationDate", "type":"DateTime", "value":"2001-01-01T10:00:00+10:00"}
{"tag": "IVCounterNonce", "type":"ByteString", "value":"a1b2c3d4"}
{"tag": "PrivateKeyTemplateAttribute", "type":"Structure", "value":[]}
{"tag": "0x545352", "type":"TextString", "value":"This is an extension"}
{"tag": "WELL_KNOWN_EXTENSION", "type":"TextString", "value":"This is an extension"}
```

4.1.6.2 Structure

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

```
{"tag": "ProtocolVersion", "type":"Structure", "value":[
  {"tag": "ProtocolVersionMajor", "type":"Integer", "value":1},
  {"tag": "ProtocolVersionMajor", "type":"Integer", "value":0}
]}
{"tag": "ProtocolVersion", "value":[
  {"tag": "ProtocolVersionMajor", "type":"Integer", "value":1},
  {"tag": "ProtocolVersionMajor", "type":"Integer", "value":0}
]}
```

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

4.1.6.3 Integer

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

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

4.1.6.4 Integer - Special case for Masks

(Cryptographic Usage Mask, Storage Status Mask):

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

```
{"tag": "CryptographicUsageMask", "type":"Integer", "value": "0x0000100c"}
{"tag": "CryptographicUsageMask", "type":"Integer", "value": "Encrypt|Decrypt|CertificateSign"}
{"tag": "CryptographicUsageMask", "type":"Integer", "value":
"CertificateSign|0x00000004|0x0000008"}
{"tag": "CryptographicUsageMask", "type":"Integer", "value": "CertificateSign|0x0000000c"}
```

4.1.6.5 Long Integer

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

```
{"tag": "0x540001", "type":"LongInteger", "value":"0xfffffffffffffffe"}
{"tag": "0x540001", "type":"LongInteger", "value":-2}
{"tag": "UsageLimitsCount", "type":"LongInteger", "value":"0x1000000000000000"}
```

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

4.1.6.6 Big Integer

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

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

4.1.6.7 Enumeration

For JSON, value may contain:

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

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

4.1.6.8 Boolean

For JSON, value must be either a hex string, or a JSON Boolean ‘true’ or ‘false’.

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

4.1.6.9 Text String

For JSON, value must be a String

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

4.1.6.10 Byte String

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

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

4.1.6.11 Date-Time

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

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

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

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

4.1.6.12 Interval

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

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

5 JSON Profile Test Cases

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

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

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

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

```
# 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>

420078010000009042007701000000484200690100000020420 06a020000000400000001000000004200 6b020000000400000000000000000420050020000000400000100000000004200 0d0200000004000000010000000042000f01000000384200 5c050000000400000018000000004200790100000020420074050000000400000001000000004200740500000004000000020000000

{"tag":"RequestMessage", "value":[
  {"tag":"RequestHeader", "value":[
    {"tag":"ProtocolVersion", "value":[
      {"tag":"ProtocolVersionMajor", "type":"Integer", "value":"0x00000001"},
      {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000000"}
    ]},
    {"tag":"MaximumResponseSize", "type":"Integer", "value":"0x00000100"},
    {"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"}
    ]}
  ]}
]
```

```
<ResponseMessage>
  <ResponseHeader>
    <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>

      42007b01000000a042007a0100000048420069010000002042006a020000000400000001000000000
      42006b020000000400000000000000000420092090000000800000000051caafbd42000d0200000004
      000000010000000042000f0100000048420005c050000000400000018000000004200 7f0500000004
      000000010000000042007e0500000004000000020000000042007d0700000009544f4f5f4c415247
      4500000000000000

      {"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"}
        ]}
      ]}
```

```
      # 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>

      420078010000009042007701000000484200690100000020420 06a020000000400000001000000000
      42006b020000000400000000000000000420050020000000400000800000000000420 00d0200000004
      000000010000000042000f0100000038420 05c050000000400000018000000004200790100000020
      420074050000000400000001000000004200740500000004000000020000000 0

      {"tag":"RequestMessage", "value":[
        {"tag":"RequestHeader", "value":[
          {"tag":"ProtocolVersion", "value":[
```

```
      {"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"}
    ]}
  ]}
]}
```

```
<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="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"/>
      <Operation type="Enumeration" value="CreateKeyPair"/>
      <Operation type="Enumeration" value="ReKey"/>
      <Operation type="Enumeration" value="Archive"/>
      <Operation type="Enumeration" value="Recover"/>
      <Operation type="Enumeration" value="ObtainLease"/>
      <Operation type="Enumeration" value="Certify"/>
      <Operation type="Enumeration" value="ReCertify"/>
      <Operation type="Enumeration" value="Notify"/>
      <Operation type="Enumeration" value="Put"/>
      <ObjectType type="Enumeration" value="Certificate"/>
      <ObjectType type="Enumeration" value="SymmetricKey"/>
      <ObjectType type="Enumeration" value="SecretData"/>
      <ObjectType type="Enumeration" value="PublicKey"/>
      <ObjectType type="Enumeration" value="PrivateKey"/>
      <ObjectType type="Enumeration" value="Template"/>
```

```
      <ObjectType type="Enumeration" value="OpaqueObject"/>
      <ObjectType type="Enumeration" value="SplitKey"/>
    </ResponsePayload>
  </BatchItem>
</ResponseMessage>

42007b01000002a042007a0100000048420069010000002042006a020000000400000001000000000
42006b020000000400000000000000004200920900000008000000051caafbd42000d0200000004
000000010000000042000f01000002484200 5c0500000004000000180000000042007f0500000004
000000000000000042007c01000002204200 5c0500000004000000180000000042005c0500000004
000000080000000042005c0500000004000000140000000042005c0500000004000000a00000000
42005c0500000004000000010000000042005c0500000004000000030000000042005c0500000004
0000000b0000000042005c0500000004000000c0000000042005c0500000004000000d00000000
42005c0500000004000000e0000000042005c0500000004000000f0000000042005c0500000004
000000120000000042005c0500000004000000130000000042005c0500000004000000 1a00000000
42005c0500000004000000190000000042005c0500000004000000090000000042005c0500000004
000000110000000042005c0500000004000000020000000042005c0500000004000000400000000
42005c0500000004000000150000000042005c0500000004000000160000000042005c0500000004
000000100000000042005c0500000004000000060000000042005c0500000004000000070000000
42005c0500000004000000 1b0000000042005c0500000004000000 1c0000000420057050000000 4
000000010000000420057050000000400000002000000042005705000000040000000700000000
42005705000000040000000300000000420057050000000400000004000000042005705000000 4
000000060000000420057050000000400000008000000042005705000000040000005000000000

{"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

6 XML Profile

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

6.1 XML Encoding

6.1.1 Hex representations

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

6.1.2 Tags

Tags are a String that may contain either:

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

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

6.1.3 Normalizing Names

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

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

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

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

Example python name normalization code

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

Example perl name normalization code

6.1.4 Type

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

- Structure
- Integer
- LongInteger
- BigInteger
- Enumeration
- Boolean
- TextString
- ByteString
- DateTime
- Interval

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

6.1.5 Value

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

6.1.6 XML Element Encoding

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

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

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

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

urn:oasis:tc:kmip:xmlns

6.1.6.1 Tags

Tags are a String that may contain either:

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

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

```
<ActivationDate xmlns="urn:oasis:tc:kmip:xmlns" type="DateTime" value="2001-01-
01T10:00:00+10:00"/>
<IVCounterNonce type="ByteString" value="a1b2c3d4"/>
<PrivateKeyTemplateAttribute type="Structure"/>
<TTLV tag="0x545352" name="SomeExtension" type="TextString" value="This is an extension"/>
<WELL_KNOWN_EXTENSION type="TextString" value="This is an extension"/>
```

6.1.6.2 Structure

For XML, sub-items are nested elements.

```
<ProtocolVersion type="Structure">
  <ProtocolVersionMajor type="Integer" value="1"/>
  <ProtocolVersionMinor type="Integer" value="0"/>
</ProtocolVersion>
<ProtocolVersion>
  <ProtocolVersionMajor type="Integer" value="1"/>
  <ProtocolVersionMinor type="Integer" value="0"/>
</ProtocolVersion>
```

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

6.1.6.3 Integer

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

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

6.1.6.4 Integer - Special case for Masks

(Cryptographic Usage Mask, Storage Status Mask):

Integer mask values can also be encoded as a String containing mask components. XML uses an attribute with XML type xsd:list which uses a space separator. Components may be either the text of the 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 string.

```
<CryptographicUsageMask type="Integer" value="0x0000100c"/>
<CryptographicUsageMask type="Integer" value="Encrypt Decrypt CertificateSign"/>
<CryptographicUsageMask type="Integer" value="CertificateSign 0x00000004 0x0000008"/>
<CryptographicUsageMask type="Integer" value="CertificateSign 0x0000000c"/>
```

6.1.6.5 Long Integer

For XML, value uses XML schema type xsd:long

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

6.1.6.6 Big Integer

For XML, value uses XML schema type xsd:hexBinary

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

6.1.6.7 Enumeration

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

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

6.1.6.8 Boolean

For XML, value uses XML schema type xsd:Boolean

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

6.1.6.9 Text String

XML uses schema type xsd:string

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

6.1.6.10 Byte String

XML uses schema type xsd:hexBinary

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

6.1.6.11 Date-Time

For XML, value uses schema type xsd:dateTime

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

6.1.6.12 Interval

XML uses schema type xsd:unsignedInt

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

7 XML Profile Test Cases

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

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

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

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>

      420078010000009042007701000000484200690100000020420060a020000000400000001000000000
      42006b020000000400000000000000000420050020000000400000100000000000420000d0200000004
      000000010000000042000f0100000038420050c05000000040000001800000000420079010000002 0
      420074050000000400000001000000004200740500000004000000020000000 0
```

```
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>

      42007b01000000a042007a0100000048420069010000002042006a020000000400000001000000000
      42006b020000000400000000000000000420092090000000800000000051caafbd42000d0200000004
      000000010000000042000f0100000048420050c05000000040000001800000000420070f0500000004
```

```
0000000100000000042007e0500000004000000020000000042007d0700000009544f4f5f4c415247
4500000000000000
```

```
# 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>

420078010000009042007701000000484200690100000020420 06a020000000400000001000000004200 6b020000000400000000000000004200500200000000400000800000000000420 00d0200000004000000010000000042000f0100000038420 05c0500000004000000180000000042007901000000204200740500000004000000010000000042007405000000040000000200000000
```

```
<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="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"/>
```

```
      <Operation type="Enumeration" value="CreateKeyPair"/>
      <Operation type="Enumeration" value="ReKey"/>
      <Operation type="Enumeration" value="Archive"/>
      <Operation type="Enumeration" value="Recover"/>
      <Operation type="Enumeration" value="ObtainLease"/>
      <Operation type="Enumeration" value="Certify"/>
      <Operation type="Enumeration" value="ReCertify"/>
      <Operation type="Enumeration" value="Notify"/>
      <Operation type="Enumeration" value="Put"/>
      <ObjectType type="Enumeration" value="Certificate"/>
      <ObjectType type="Enumeration" value="SymmetricKey"/>
      <ObjectType type="Enumeration" value="SecretData"/>
      <ObjectType type="Enumeration" value="PublicKey"/>
      <ObjectType type="Enumeration" value="PrivateKey"/>
      <ObjectType type="Enumeration" value="Template"/>
      <ObjectType type="Enumeration" value="OpaqueObject"/>
      <ObjectType type="Enumeration" value="SplitKey"/>
    </ResponsePayload>
  </BatchItem>
</ResponseMessage>

42007b01000002a042007a01000000484200690100000020420 06a020000000400000001000000004
42006b020000000400000000000000004200920900000008000000051caafbd42000d0200000004
0000000100000000042000f0100000248420 05c0500000004000000180000000042007f0500000004
0000000000000000042007c0100000220420 05c0500000004000000180000000042005c0500000004
0000000800000000042005c0500000004000000140000000042005c0500000004000000a00000000
42005c0500000004000000010000000042005c0500000004000000030000000042005c0500000004
0000000b0000000042005c0500000004000000c0000000042005c0500000004000000d00000000
42005c0500000004000000e0000000042005c0500000004000000f0000000042005c0500000004
0000001200000000042005c0500000004000000130000000042005c0500000004000000 1a00000000
42005c0500000004000000190000000042005c0500000004000000090000000042005c0500000004
0000001100000000042005c0500000004000000020000000042005c0500000004000000400000000
42005c0500000004000000150000000042005c0500000004000000160000000042005c0500000004
0000001000000000042005c0500000004000000060000000042005c0500000004000000700000000
42005c0500000004000000 1b0000000042005c0500000004000000 1c0000000042005705000000004
0000001000000004200570500000004000000020000000042005705000000040000007000000000
4200570500000004000000030000000042005705000000040000000400000000420057050000000 4
0000006000000004200570500000004000000080000000042005705000000040000005000000000
```

480

8 Conformance

8.1 HTTPS Profile Conformance

KMIP client and server implementations conformant to this profile:

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

8.2 JSON Profile Conformance

KMIP client and server implementations conformant to this profile:

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

8.3 XML Profile Conformance

KMIP client and server implementations conformant to this profile:

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

8.4 Permitted Test Case Variations

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

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

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

8.4.1 Variable Items

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

1. UniqueIdentifier
2. PrivateKeyUniqueIdentifier
3. PublicKeyUniqueIdentifier

4. UniqueBatchItemIdentifier
5. AsynchronousCorrelationValue
6. TimeStamp
7. KeyValue / KeyMaterial including:
    a. key material content returned for managed cryptographic objects which are generated by the server
    b. wrapped versions of keys where the wrapping key is dynamic or the wrapping contains variable output for each wrap operation
8. For response containing the output of cryptographic operation in Data / SignatureData/ MACData / IVCounterNonce where:
    a. the managed object is generated by the server; or
    b. the operation inherently contains variable output
9. For the following DateTime attributes where the value is not specified in the request as a fixed DateTime value:
    a. ActivationDate
    b. ArchiveDate
    c. CompromiseDate
    d. CompromiseOccurrenceDate
    e. DeactivationDate
    f. DestroyDate
    g. InitialDate
    h. LastChangeDate
    i. ProtectStartDate
    j. ProcessStopDate
    k. ValidityDate
    l. OriginalCreationDate
10. LinkedObjectIdentifier
11. DigestValue
    a. For those managed cryptographic objects which are dynamically generated
12. KeyFormatType
    a. The key format type selected by the server when it creates managed objects
13. Digest
    a. The HashingAlgorithm selected by the server when it calculates the digest for a managed object for which it has access to the key material
    b. The Digest Value
14. Extensions reported in Query for ExtensionList and ExtensionMap
15. Application Namespaces reported in Query
16. Object Types reported in Query other than those noted as required in this profile
17. Operation Types reported in Query other than those noted as required in this profile (or any referenced profile documents)
18. For TextString attribute values containing test identifiers:
    a. Additional vendor or application prefixes
19. Additional attributes beyond those noted in the response

An implementation conformant to this Profile MAY allow the following response variations:

1. Object Group values – May or may not return one or more Object Group values not included in the requests
2. y-CustomAttributes – May or may not include additional server-specific associated attributes not included in requests
3. Message Extensions – May or may not include additional (non-critical) vendor extensions
4. TemplateAttribute – May or may not be included in responses where the Template Attribute response is noted as optional in [KMIP-SPEC]
5. AttributeIndex – May or may not include Attribute Index value where the Attribute Index value is 0 for Protocol Versions 1.1 and above.
6. ResultMessage – May or may not be included in responses and the value (if included) may vary from the text contained within the test case.
7. The list of Protocol Versions returned in a DiscoverVersion response may include additional protocol versions if the request has not specified a list of client supported Protocol Versions.
8. VendorIdentification - The value (if included) may vary from the text contained within the test case.

8.4.2 Variable behavior

An implementation conformant to this Profile SHALL allow variation of the following behavior:

1. A test may omit the clean-up requests and responses (containing Revoke and/or Destroy) at the end of the test provided there is a separate mechanism to remove the created objects during testing.
2. A test may omit the test identifiers if the client is unable to include them in requests. This includes the following attributes:
    a. Name; and
    b. x-ID

Appendix A. Acknowledgments

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

Original HTTPS Profile Proposal:

Alan Frindell, SafeNet, Inc.

Original HTTPS Profile Contributors:

Mathias Björkqvist, IBM

Participants:

Hal Aldridge, Sypris Electronics
Mike Allen, Symantec
Gordon Arnold, IBM
Todd Arnold, IBM
Richard Austin, Hewlett-Packard
Lars Bagnert, PrimeKey
Elaine Barker, NIST
Peter Bartok, Venafi, Inc.
Tom Benjamin, IBM
Anthony Berglas, Cryptsoft
Mathias Björkqvist, IBM
Kevin Bocket, Venafi
Anne Bolgert, IBM
Alan Brown, Thales e-Security
Tim Bruce, CA Technologies
Chris Burchett, Credant Technologies, Inc.
Kelley Burgin, National Security Agency
Robert Burns, Thales e-Security
Chuck Castleton, Venafi
Kenli Chong, QuintessenceLabs
John Clark, Hewlett-Packard
Tom Clifford, Symantec Corp.
Doron Cohen, SafeNet, Inc
Tony Cox, Cryptsoft
Russell Dietz, SafeNet, Inc
Graydon Dodson, Lexmark International Inc.
Vinod Duggirala, EMC Corporation
Chris Dunn, SafeNet, Inc.
Michael Duren, Sypris Electronics
James Dzierzanowski, American Express CCoE
Faisal Faruqui, Thales e-Security
Stan Feather, Hewlett-Packard
David Finkelstein, Symantec Corp.
James Fitzgerald, SafeNet, Inc.
Indra Fitzgerald, Hewlett-Packard
Judith Furlong, EMC Corporation
Susan Gleeson, Oracle
Robert Griffin, EMC Corporation
Paul Grojean, Individual
Robert Haas, IBM
Thomas Hardjono, M.I.T.
ChengDong He, Huawei Technologies Co., Ltd.
Steve He, Vormetric

Kurt Heberlein, Hewlett-Packard
Larry Hofer, Emulex Corporation
Maryann Hondo, IBM
Walt Hubis, NetApp
Tim Hudson, Cryptsoft
Jonas Iggbom, Venafi, Inc.
Sitaram Inguva, American Express CCoE
Jay Jacobs, Target Corporation
Glen Jaquette, IBM
Mahadev Karadiguddi, NetApp
Greg Kazmierczak, Wave Systems Corp.
Marc Kenig, SafeNet, Inc.
Mark Knight, Thales e-Security
Kathy Kriese, Symantec Corporation
Mark Lambiase, SecureAuth
John Leiseboer, Quintenssence Labs
Hal Lockhart, Oracle Corporation
Robert Lockhart, Thales e-Security
Anne Luk, Cryptsoft
Sairam Manidi, Freescale
Luther Martin, Voltage Security
Neil McEvoy, iFOSSF
Marina Milshtein, Individual
Dale Moberg, Axway Software
Jishnu Mukeri, Hewlett-Packard
Bryan Olson, Hewlett-Packard
John Peck, IBM
Rob Philpott, EMC Corporation
Denis Pochuev, SafeNet, Inc.
Reid Poole, Venafi, Inc.
Ajai Puri, SafeNet, Inc.
Saravanan Ramalingam, Thales e-Security
Peter Reed, SafeNet, Inc.
Bruce Rich, IBM
Christina Richards, American Express CCoE
Warren Robbins, Dell
Peter Robinson, EMC Corporation
Scott Rotondo, Oracle
Saikat Saha, SafeNet, Inc.
Anil Saldhana, Red Hat
Subhash Sankuratripati, NetApp
Boris Schumperli, Cryptomathic
Greg Singh, QuintessenceLabs
David Smith, Venafi, Inc
Brian Spector, Certivox
Terence Spies, Voltage Security
Deborah Steckroth, RouteOne LLC
Michael Stevens, QuintessenceLabs
Marcus Streets, Thales e-Security
Satish Sundar, IBM
Kiran Thota, VMware
Somanchi Trinath, Freescale Semiconductor, Inc.
Nathan Turajski, Thales e-Security
Sean Turner, IECA, Inc.
Paul Turner, Venafi, Inc.
Rod Wideman, Quantum Corporation
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 Zelechoski, Election Systems & Software

Appendix B. KMIP Specification Cross Reference

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

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

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

Reference Term	KMIP 1.0	KMIP 1.1	KMIP 1.2
Message Structure	7.1.	7.1.	7.1.
Operations	7.2.	7.2.	7.2.
8 Authentication			
Authentication	8	8	8
9 Message Encoding			
Alternative Name Type Enumeration	-	-	9.1.3.2.34.
Attestation Type Enumeration	-	-	9.1.3.2.36.
Batch Error Continuation Option Enumeration	9.1.3.2.29.	9.1.3.2.30.	9.1.3.2.30.
Bit Masks	9.1.3.3.	9.1.3.3.	9.1.3.3.
Block Cipher Mode Enumeration	9.1.3.2.13.	9.1.3.2.14.	9.1.3.2.14.
Cancellation Result Enumeration	9.1.3.2.24.	9.1.3.2.25.	9.1.3.2.25.
Certificate Request Type Enumeration	9.1.3.2.21.	9.1.3.2.22.	9.1.3.2.22.
Certificate Type Enumeration	9.1.3.2.6.	9.1.3.2.6.	9.1.3.2.6.
Credential Type Enumeration	9.1.3.2.1.	9.1.3.2.1.	9.1.3.2.1.
Cryptographic Algorithm Enumeration	9.1.3.2.12.	9.1.3.2.13.	9.1.3.2.13.
Cryptographic Usage Mask	9.1.3.3.1.	9.1.3.3.1.	9.1.3.3.1.
Defined Values	9.1.3.	9.1.3.	9.1.3.
Derivation Method Enumeration	9.1.3.2.20.	9.1.3.2.21.	9.1.3.2.21.
Digital Signature Algorithm Enumeration	-	9.1.3.2.7.	9.1.3.2.7.
Encoding Option Enumeration	-	9.1.3.2.32.	9.1.3.2.32.
Enumerations	9.1.3.2.	9.1.3.2.	9.1.3.2.
Examples	9.1.2.	9.1.2.	9.1.2.
Hashing Algorithm Enumeration	9.1.3.2.15.	9.1.3.2.16.	9.1.3.2.16.
Item Length	9.1.1.3.	9.1.1.3.	9.1.1.3.
Item Tag	9.1.1.1.	9.1.1.1.	9.1.1.1.
Item Type	9.1.1.2.	9.1.1.2.	9.1.1.2.
Item Value	9.1.1.4.	9.1.1.4.	9.1.1.4.
Key Compression Type Enumeration	9.1.3.2.2.	9.1.3.2.2.	9.1.3.2.2.
Key Format Type Enumeration	9.1.3.2.3.	9.1.3.2.3.	9.1.3.2.3.
Key Role Type Enumeration	9.1.3.2.16.	9.1.3.2.17.	9.1.3.2.17.
Key Value Location Type Enumeration	-	-	9.1.3.2.35.
Link Type Enumeration	9.1.3.2.19.	9.1.3.2.20.	9.1.3.2.20.
Name Type Enumeration	9.1.3.2.10.	9.1.3.2.11.	9.1.3.2.11.
Object Group Member Enumeration	-	9.1.3.2.33.	9.1.3.2.33.
Object Type Enumeration	9.1.3.2.11.	9.1.3.2.12.	9.1.3.2.12.
Opaque Data Type Enumeration	9.1.3.2.9.	9.1.3.2.10.	9.1.3.2.10.
Operation Enumeration	9.1.3.2.26.	9.1.3.2.27.	9.1.3.2.27.
Padding Method Enumeration	9.1.3.2.14.	9.1.3.2.15.	9.1.3.2.15.
Put Function Enumeration	9.1.3.2.25.	9.1.3.2.26.	9.1.3.2.26.
Query Function Enumeration	9.1.3.2.23.	9.1.3.2.24.	9.1.3.2.24.
Recommended Curve Enumeration for ECDSA, ECDH, and ECMQV	9.1.3.2.5.	9.1.3.2.5.	9.1.3.2.5.
Result Reason Enumeration	9.1.3.2.28.	9.1.3.2.29.	9.1.3.2.29.
Result Status Enumeration	9.1.3.2.27.	9.1.3.2.28.	9.1.3.2.28.
Revocation Reason Code Enumeration	9.1.3.2.18.	9.1.3.2.19.	9.1.3.2.19.
Secret Data Type Enumeration	9.1.3.2.8.	9.1.3.2.9.	9.1.3.2.9.
Split Key Method Enumeration	9.1.3.2.7.	9.1.3.2.8.	9.1.3.2.8.
State Enumeration	9.1.3.2.17.	9.1.3.2.18.	9.1.3.2.18.
Storage Status Mask	9.1.3.3.2.	9.1.3.3.2.	9.1.3.3.2.
Tags	9.1.3.1.	9.1.3.1.	9.1.3.1.
TTLV Encoding	9.1.	9.1.	9.1.
TTLV Encoding Fields	9.1.1.	9.1.1.	9.1.1.
Usage Limits Unit Enumeration	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**
Validity Indicator Enumeration	9.1.3.2.22.	9.1.3.2.23.	9.1.3.2.23.
Wrapping Method Enumeration	9.1.3.2.4.	9.1.3.2.4.	9.1.3.2.4.
XML Encoding	9.2.	-	-
10 Transport			
Transport	10	10	10
12 KMIP Server and Client Implementation Conformance			
Conformance clauses for a KMIP Server	12.1.	-	-
KMIP Client Implementation Conformance	-	12.2.	12.2.
KMIP Server Implementation Conformance	-	12.1.	12.1.

Appendix C. Revision History

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