Key Management Interoperability Protocol Profiles Version 1.3

Committee Specification Draft 01

07 April 2016

Specification URIs

This version:

http://docs.oasis-open.org/kmip/profiles/v1.3/csd01/kmip-profiles-v1.3-csd01.doc (Authoritative)
http://docs.oasis-open.org/kmip/profiles/v1.3/csd01/kmip-profiles-v1.3-csd01.html
http://docs.oasis-open.org/kmip/profiles/v1.3/csd01/kmip-profiles-v1.3-csd01.pdf

Previous version:

N/A

Latest version:

http://docs.oasis-open.org/kmip/profiles/v1.3/kmip-profiles-v1.3.doc (Authoritative)
http://docs.oasis-open.org/kmip/profiles/v1.3/kmip-profiles-v1.3.html
http://docs.oasis-open.org/kmip/profiles/v1.3/kmip-profiles-v1.3.pdf

Technical Committee:

OASIS Key Management Interoperability Protocol (KMIP) TC

Chairs:

Saikat Saha (saikat.saha@oracle.com), Oracle
Tony Cox (tjc@cryptsoft.com), Cryptsoft Pty Ltd.

Editors:

Tim Hudson (tjh@cryptsoft.com), Cryptsoft Pty Ltd.
Robert Lockhart (Robert.Lockhart@thalesesec.com), Thales e-Security

Additional artifacts:

This prose specification is one component of a Work Product that also includes

- Test cases: http://docs.oasis-open.org/kmip/profiles/v1.3/csd01/test-cases/kmip-v1.3/mandatory/ and http://docs.oasis-open.org/kmip/profiles/v1.3/csd01/test-cases/kmip-v1.3/optional/

Related work:

This specification replaces or supersedes:

- *Key Management Interoperability Protocol Profiles Version 1.0.* Edited by Robert Griffin and Subhash Sankuratripati. Latest version: http://docs.oasis-open.org/kmip/profiles/v1.0/kmip-profiles-1.0.html.
- *Key Management Interoperability Protocol Profiles Version 1.1*.Edited by Robert Griffin and Subhash Sankuratripati. Latest version: http://docs.oasis-open.org/kmip/profiles/v1.1/kmip-profiles-v1.1.html.
- *Key Management Interoperability Protocol Profiles Version 1.2.* Edited by Tim Hudson and Robert Lockhart. Latest version: http://docs.oasis-open.org/kmip/profiles/v1.2/kmip-profiles-v1.2.html.

This specification is related to:

- *Key Management Interoperability Protocol Specification Version 1.3*. Edited by Kiran Thota and Tony Cox. Latest version: http://docs.oasis-open.org/kmip/spec/v1.3/kmip-spec-v1.3.html.
- *Key Management Interoperability Protocol Test Cases Version 1.3*. Edited by Tim Hudson and Mark Joseph. Latest version: http://docs.oasis-open.org/kmip/testcases/v1.3/kmip-testcases-v1.3.html.
- *Key Management Interoperability Protocol Usage Guide Version 1.3*. Edited by Judith Furlong. Latest version: http://docs.oasis-open.org/kmip/ug/v1.3/kmip-ug-v1.3.html.

Abstract:

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

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. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=kmip#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://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 TC's web page (https://www.oasis-open.org/committees/kmip/ipr.php).

Citation format:

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

[KMIP-Profiles-v1.3]

Key Management Interoperability Protocol Profiles Version 1.3. Edited by Tim Hudson and Robert Lockhart. 07 April 2016. OASIS Committee Specification Draft 01. http://docs.oasis-open.org/kmip/profiles/v1.3/csd01/kmip-profiles-v1.3-csd01.html. Latest version: http://docs.oasis-open.org/kmip/profiles/v1.3/kmip-profiles-v1.3.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 https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

1 Introduction

This document specifies conformance clauses in accordance with the OASIS TC Process ([TC-PROC] section 2.18 paragraph 8a) for the KMIP Specification ([KMIP-SPEC] 12.1 and 12.2) for a KMIP server or KMIP client through profiles that define the use of KMIP objects, attributes, operations, message elements and authentication methods within specific contexts of KMIP server and client interaction.

These profiles define a set of normative constraints for employing KMIP within a particular environment or context of use. They may, optionally, require the use of specific KMIP functionality or in other respects define the processing rules to be followed by profile actors.

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

[KMIP-SPEC] *Key Management Interoperability Protocol Specification Version 1.3.* Edited by Kiran Thota and Tony Cox. Latest version: http://docs.oasis-open.org/kmip/spec/v1.3/kmip-spec-v1.3.html.

[SuiteB] *Suite B Cryptography / Cryptographic Interoperability*, http://www.nsa.gov/ia/programs/suiteb_cryptography/

[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.

[RFC2246] T. Dierks & C.Allen, *The TLS Protocol, Version 1.0***,** http://www.ietf.org/rfc/rfc2246.txt**,** IETF RFC 2246, January 1999

[RFC2818] E. Rescorla, *HTTP over TLS***,** IETF RFC 2818, May 2000, http://www.ietf.org/rfc/rfc2818.txt

[RFC3268] P. Chown, *Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security* (TLS), http://www.ietf.org/rfc/rfc3268.txt, IETF RFC 3268, June 2002

[RFC4346] T. Dierks & E. Rescorla, *The Transport Layer Security (TLS) Protocol, Version 1.1*, http://www.ietf.org/rfc/rfc4346.txt, IETF RFC 4346, April 2006

[RFC5246] T. Dierks & E. Rescorla, *The Transport Layer Security (TLS) Protocol, Version 1.2*, http://www.ietf.org/rfc/rfc5246.txt, IETF RFC 5246, August 2008

[RFC7159] Bray, T., Ed., *The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 7159, March 2014. http://www.ietf.org/rfc/rfc7159.txt

[CNSSP-15] N.S.A., “National Information Assurance Policy on the Use of Public Standards for the Secure Sharing of Information Among National Security Systems”, 1 October 2012, https://www.cnss.gov/CNSS/issuances/Policies.cfm.

[XML] Bray, Tim, et.al. eds, Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008, available at http://www.w3.org/TR/2008/REC-xml-20081126/

1.3 Non-Normative References

[TC-PROC] *OASIS TC Process.* 1 May 2014. OASIS Process. https://www.oasis-open.org/policies-guidelines/tc-process.

[XML-SCHEMA] Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 26 November 2008, available at http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

2 Profiles

This document defines a list of KMIP Profiles. A profile may be standalone or may be specified in terms of changes relative to another profile.

2.1 Profile Requirements

The following items SHALL be addressed by each profile.

1. Specify the versions of the KMIP specification (protocol versions) that SHALL be supported
2. Specify the list of objects that SHALL be supported
3. Specify the list of Authentication Suites that SHALL be supported
4. Specify the list of Attributes that SHALL be supported
5. Specify the list of Operations that SHALL be supported
6. Specify any additional message content that SHALL be supported
7. Specify any other requirements that SHALL be supported
8. For profiles other than the Baseline Client, Baseline Server and Complete Server the profile SHALL specify the mandatory test cases that SHALL be supported and MAY specify the optional test cases that MAY be supported by conforming implementations

2.2 Guidelines for other Profiles

Any vendor or organization, such as other standards bodies, MAY create a KMIP Profile and publish it.

1. The profile SHALL be publicly available.
2. The KMIP Technical Committee SHALL be formally advised of the availability of the profile and the location of the published profile.
3. The profile SHALL meet all the requirements of section 2.1
4. The KMIP Technical Committee SHOULD review the profile prior to publication.

3 Authentication Suites

This section contains the list of the channel security, channel options, and server and client authentication requirements for a KMIP profile. Other Authentication Suites MAY be defined for other KMIP Profiles.

An Authentication Suite provides at least the following:

1. All communication over the security channel SHALL provide confidentiality and integrity
2. All communication over the security channel SHALL provide assurance of server authenticity
3. All communication over the security channel for Operations other than `Query` and `Discover Versions` SHALL provide assurance of client authenticity
4. All options such as channel protocol version and cipher suites for the secuity channel SHALL be specified

3.1 Basic Authentication Suite

This authentication suite stipulates that a profile conforming to the Basic Authentication Suite SHALL use TLS to negotiate a secure channel.

3.1.1 Basic Authentication Protocols

Conformant KMIP clients or servers SHALL support:

- `TLS v1.0 [RFC2246] and [RFC3268]`

Conformant KMIP clients or servers SHOULD support:

- `TLS v1.2 [RFC5246]`

Conformant KMIP clients or servers MAY support:

- `TLS v1.1 [RFC4346]`

Conformant KMIP clients or servers SHALL NOT support:

- `SSL v3.0`
- `SSL v2.0`
- `SSL v1.0`

3.1.2 Basic Authentication Cipher Suites

Conformant KMIP clients or servers SHALL support the following cipher suites:

- `TLS_RSA_WITH_AES_128_CBC_SHA`

Conformant KMIP clients and servers MAY support the following cipher suites:

- `TLS_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_256_CBC_SHA256`
- `TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA`
- `TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
- `TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA`
- `TLS_DH_DSS_WITH_AES_128_CBC_SHA`
- `TLS_DH_RSA_WITH_AES_128_CBC_SHA`
- `TLS_DHE_DSS_WITH_AES_128_CBC_SHA`
- `TLS_DHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_DH_DSS_WITH_AES_256_CBC_SHA`

- TLS_DH_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DH_DSS_WITH_AES_128_CBC_SHA256
- TLS_DH_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DH_DSS_WITH_AES_256_CBC_SHA256
- TLS_DH_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_PSK_WITH_3DES_EDE_CBC_SHA
- TLS_PSK_WITH_AES_128_CBC_SHA
- TLS_PSK_WITH_AES_256_CBC_SHA
- TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_PSK_WITH_AES_128_CBC_SHA
- TLS_DHE_PSK_WITH_AES_256_CBC_SHA
- TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_PSK_WITH_AES_128_CBC_SHA
- TLS_RSA_PSK_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

Conformant KMIP clients or servers SHALL NOT support any cipher suite not listed above.

NOTE: TLS 1.0 has known security issues and implementations that need protections against known issues SHOULD considering using the TLS 1.2 Authentication Suite (3.2)

3.1.3 Basic Authentication Client Authenticity

Conformant KMIP servers SHALL require the use of channel (TLS) mutual authentication to provide assurance of client authenticity for all operations other than:

- Query
- Discover Versions

Conformant KMIP servers SHALL use the identity derived from the channel mutual authentication to determine the client identity if the KMIP client requests do not contain an Authentication object.

Conformant KMIP servers SHALL use the identity derived from the channel mutual authentication along with the Credential information to determine the client identity if the KMIP client requests contain an Authentication object.

The exact mechanisms determining the client identity are outside the scope of this specification.

3.1.4 Basic Authentication KMIP Port Number

Conformant KMIP servers SHALL use TCP port number 5696, as assigned by IANA.

3.2 TLS 1.2 Authentication Suite

This authentication suite stipulates that a profile conforming to the TLS 1.2 Authentication Suite SHALL use TLS version 1.2 to negotiate a secure channel.

3.2.1 TLS 1.2 Authentication Protocols

Conformant KMIP clients and servers SHALL support:

- `TLS v1.2 [RFC2246]`

3.2.2 TLS 1.2 Authentication Cipher Suites

Conformant KMIP servers SHALL support the following cipher suites:

- `TLS_RSA_WITH_AES_256_CBC_SHA256`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`

Conformant KMIP servers and clients MAY support the cipher suites specified as MAY in Basic Authentication Cipher Suites (3.1.2) of the Basic Authentication Suite.

3.2.3 TLS 1.2 Authentication Client Authenticity

Conformant KMIP servers and clients SHALL handle client authenticity in accordance with Basic Authentication Client Authenticity (3.1.3) of the Basic Authentication Suite

3.2.4 TLS 1.2 Authentication KMIP Port Number

Conformant KMIP servers and clients SHALL handle the KMIP port number in in accordance with Basic Authentication KMIP Port Number (3.1.4) of the Basic Authentication Suite.

3.3 Suite B minLOS_128 Authentication Suite

Implementations conformant to this profile SHALL use TLS to negotiate a mutually-authenticated connection.

3.3.1 Suite B minLOS_128 Protocols

Conformant KMIP clients and servers SHALL support:

- TLS v1.2 [RFC5246]

3.3.2 Suite B minLOS_128 Cipher Suites

Conformant KMIP servers SHALL support the following cipher suites:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

3.3.3 Suite B minLOS_128 Client Authenticity

Conformant KMIP servers and clients SHALL handle client authenticity in accordance with TLS 1.2 Authentication Client Authenticity (3.2.3).

3.3.4 Suite B minLOS_128 KMIP Port Number

Conformant KMIP servers and clients SHALL handle the KMIP port number in accordance with TLS 1.2 Authentication KMIP Port Number (3.2.4).

3.4 Suite B minLOS_192 Authentication Suite

Implementations conformant to this profile SHALL use TLS to negotiate a mutually-authenticated connection.

3.4.1 Suite B minLOS_192 Protocols

Conformant KMIP clients and servers SHALL support:

- TLS v1.2 [RFC5246]

3.4.2 Suite B minLOS_192 Cipher Suites

Conformant KMIP servers SHALL support the following cipher suites:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

3.4.3 Suite B minLOS_192 Client Authenticity

Conformant KMIP servers and clients SHALL handle client authenticity in accordance with TLS 1.2 Authentication Client Authenticity (3.2.3).

3.4.4 Suite B minLOS_192 KMIP Port Number

Conformant KMIP servers and clients SHALL handle the KMIP port number in accordance with TLS 1.2 Authentication KMIP Port Number (3.2.4).

3.5 HTTPS Authentication Suite

This authentication suite stipulates that a profile conforming to the HTTPS Authentication Suite SHALL use HTTP over TLS [RFC2818] to negotiate a secure channel.

3.5.1 HTTPS Protocols

Conformant KMIP servers and clients SHALL handle client authenticity in accordance with Basic Authentication Protocols (3.1.1).

3.5.2 HTTPS Cipher Suites

Conformant KMIP servers and clients SHALL handle client authenticity in accordance with Basic Authentication Cipher Suites (3.1.2).

3.5.3 HTTPS Authenticity

Conformant KMIP servers and clients SHALL handle client authenticity in accordance with Basic Authentication Client Authenticity (3.1.3).

3.5.4 HTTPS 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 HTTPS and non-HTTPS 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 for HTTPS usage.

4 Conformance Test Cases

The test cases define a number of request-response pairs for KMIP operations. Each test case is provided in the XML format specified in XML Encoding (5.4.1) intended to be both human-readable and usable by automated tools.

Each test case has a unique label (the section name) which includes indication of mandatory (-M-) or optional (-O-) status and the protocol version major and minor numbers as part of the identifier.

The test cases may depend on a specific configuration of a KMIP client and server being configured in a manner consistent with the test case assumptions.

Where possible the flow of unique identifiers between tests, the date-time values, and other dynamic items are indicated using symbolic identifiers – in actual request and response messages these dynamic values will be filled in with valid values.

Symbolic identifiers are of the form $UPPERCASE_NAME followed by optional unique index value. Wherever a symbolic identifier occurs in a test cases the implementation must replace it with a reasonable appearing datum of the expected type. Time values can be specified in terms of an offset from the current time in seconds of the form $NOW or $NOW-n or $NOW+n.

Note: the values for the returned items and the custom attributes are illustrative. Actual values from a real client or server system may vary as specified in section 4.1.

4.1 Permitted Test Case Variations

Whilst the test cases provided in a 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 section SHALL be deemed non-conformant.

4.1.1 Variable Items

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

1. UniqueIdentifier
2. PrivateKeyUniqueIdentifier
3. PublicKeyUniqueIdentifier
4. UniqueBatchItemID
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 the profile
17. Operation Types reported in Query other than those noted as required in the profile
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 a 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.

4.1.2 Variable behavior

An implementation conformant to a 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
3. A test MAY perform requests with multiple batch items or as multiple requests with a single batch item provided the sequence of operations are equivalent
4. A request MAY contain an optional *Authentication* [KMIP_SPEC] structure within each request
5. The order of Attributes returned in a GetAttributes operation is not specified in [KMIP-SPEC] and an implementation MAY return the list of items in any order provided all noted items are present. Any permutation of the order of the required entries is allowed.
6. For all profiles in each BatchItem of a client request including multiple batch items (i.e. BatchCount is greater than one) the UniqueBatchItemID must be specified. UniqueBatchItemID MAY be specified for client requests containing a single batch item (BatchCount equals one) or MAY be omitted. Any reasonable appearing datum of the expected type is permitted.

5 Profiles

5.1 Base Profiles

5.1.1 Baseline Client

A Baseline Client provides some of the most basic functionality that is expected of a conformant KMIP client – the ability to request information about the server.

An implementation is a conforming Baseline Client Clause if it meets the following conditions:

1. Supports the conditions required by the KMIP Client conformance clauses ([KMIP-SPEC] 12.2)
2. Supports the following objects:
    a. Attribute ([KMIP-SPEC] 2.1.1)
    b. Template-Attribute Structure ([KMIP-SPEC] 2.1.8)
3. Supports the following subsets of attributes:
    a. Unique Identifier ([KMIP-SPEC] 3.1)
    b. Object Type ([KMIP-SPEC] 3.3)
    c. Digest ([KMIP-SPEC] 3.17)
    d. Default Operation Policy ([KMIP-SPEC] 3.18.2)
    e. State ([KMIP-SPEC] 3.22)
    f. Initial Date ([KMIP-SPEC] 3.23)
    g. Activation Date ([KMIP-SPEC] 3.24)
    h. Deactivation Date ([KMIP-SPEC] 3.27)
    i. Last Change Date ([KMIP-SPEC] 3.38)
4. Supports the ID Placeholder ([KMIP-SPEC] 4)
5. Supports the following client-to-server operations:
    a. Locate ([KMIP-SPEC] 4.9)
    b. Get ([KMIP-SPEC] 4.11)
    c. Get Attributes ([KMIP-SPEC] 4.12)
    d. Query ([KMIP-SPEC] 4.25)
6. Supports the following message contents:
    a. Protocol Version ([KMIP-SPEC] 6.1)
    b. Operation ([KMIP-SPEC] 6.2)
    c. Maximum Response Size ([KMIP-SPEC] 6.3)
    d. Unique Batch Item ID ([KMIP-SPEC] 6.4)
    e. Time Stamp ([KMIP-SPEC] 6.5)
    f. Asynchronous Indicator ([KMIP-SPEC] 6.7)
    g. Result Status ([KMIP-SPEC] 6.9)
    h. Result Reason ([KMIP-SPEC] 6.10)
    i. Batch Order Option ([KMIP-SPEC] 6.12)
    j. Batch Error Continuation Option ([KMIP-SPEC] 6.13)
    k. Batch Count ([KMIP-SPEC] 6.14)
    l. Batch Item ([KMIP-SPEC] 6.15)
1. Supports Message Format ([KMIP-SPEC] 7)
2. Supports Authentication ([KMIP-SPEC] 8)
3. Supports the TTLV encoding ([KMIP-SPEC] 9.1)
4. Supports the transport requirements ([KMIP-SPEC] 10)
5. Supports Error Handling ([KMIP-SPEC] 11) for any supported object, attribute, or operation
6. Optionally supports any clause within [KMIP-SPEC] that is not listed above.

7. Optionally supports extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements

5.1.2 Baseline Server

A Baseline Server provides the most basic functionality that is expected of a conformant KMIP server – the ability to provide information about the server and the managed objects supported by the server.

if it meets the following conditions:

1. Supports the conditions required by the KMIP Server conformance clauses ([KMIP-SPEC] 12.1)
2. Supports the following objects:
    a. Attribute ([KMIP-SPEC] 2.1.1)
    b. Credential ([KMIP-SPEC] 2.1.2)
    c. Key Block ([KMIP-SPEC] 2.1.3)
    d. Key Value ([KMIP-SPEC] 2.1.4)
    e. Template-Attribute Structure ([KMIP-SPEC] 2.1.8)
    f. Extension Information ([KMIP-SPEC] 2.1.9)
3. Supports the following subsets of attributes:
    a. Unique Identifier ([KMIP-SPEC] 3.1)
    b. Name ([KMIP-SPEC] 3.2)
    c. Object Type ([KMIP-SPEC] 3.3)
    d. Cryptographic Algorithm ([KMIP-SPEC] 3.4)
    e. Cryptographic Length ([KMIP-SPEC] 3.5)
    f. Cryptographic Parameters ([KMIP-SPEC] 3.6)
    g. Digest ([KMIP-SPEC] 3.17)
    h. Default Operation Policy ([KMIP-SPEC] 3.18.2)
    i. Cryptographic Usage Mask ([KMIP-SPEC] 3.19)
    j. State ([KMIP-SPEC] 3.22)
    k. Initial Date ([KMIP-SPEC] 3.23)
    l. Activation Date ([KMIP-SPEC] 3.24)
    m. Deactivation Date ([KMIP-SPEC] 3.27)
    n. Compromise Occurrence Date ([KMIP-SPEC] 3.29)
    o. Compromise Date ([KMIP-SPEC] 3.30)
    p. Revocation Reason ([KMIP-SPEC] 3.31)
    q. Last Change Date ([KMIP-SPEC] 3.38)
4. Supports the ID Placeholder ([KMIP-SPEC] 4)
5. Supports the following client-to-server operations:
    a. Locate ([KMIP-SPEC] 4.9)
    b. Check ([KMIP-SPEC] 4.10)
    c. Get ([KMIP-SPEC] 4.11)
    d. Get Attributes ([KMIP-SPEC] 4.12)
    e. Get Attribute List ([KMIP-SPEC] 4.13)
    f. Add Attribute ([KMIP-SPEC] 4.14)
    g. Modify Attribute ([KMIP-SPEC] 4.15)
    h. Delete Attribute ([KMIP-SPEC] 4.16)
    i. Activate ([KMIP-SPEC] 4.19)
    j. Revoke ([KMIP-SPEC] 4.20)
    k. Destroy ([KMIP-SPEC] 4.21)
    l. Query ([KMIP-SPEC] 4.25)
    m. Discover Versions ([KMIP-SPEC] 4.26)
6. Supports the following message contents:
    a. Protocol Version ([KMIP-SPEC] 6.1)
    b. Operation ([KMIP-SPEC] 6.2)
    c. Maximum Response Size ([KMIP-SPEC] 6.3)

d. Unique Batch Item ID ([KMIP-SPEC] 6.4)
e. Time Stamp ([KMIP-SPEC] 6.5)
f. Asynchronous Indicator ([KMIP-SPEC] 6.7)
g. Result Status ([KMIP-SPEC] 6.9)
h. Result Reason ([KMIP-SPEC] 6.10)
i. Batch Order Option ([KMIP-SPEC] 6.12)
j. Batch Error Continuation Option ([KMIP-SPEC] 6.13)
k. Batch Count ([KMIP-SPEC] 6.14)
l. Batch Item ([KMIP-SPEC] 6.15)
m. Attestation Capable Indicator ([KMIP-SPEC] 6.17)

7. Supports Message Format ([KMIP-SPEC] 7)
8. Supports Authentication ([KMIP-SPEC] 8)
9. Supports the TTLV encoding ([KMIP-SPEC] 9.1)
10. Supports the transport requirements ([KMIP-SPEC] 10)
11. Supports Error Handling ([KMIP-SPEC] 11) for any supported object, attribute, or operation
12. Optionally supports any clause within [KMIP-SPEC] that is not listed above
13. Optionally supports extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements

5.2 Complete Server Profile

A Complete Server provides functionality that is expected of a conformant KMIP server that implements the entire specification.

An implementation is a conforming Complete Server if it meets the following conditions:

1. Supports the conditions required by the KMIP Server conformance clauses ([KMIP-SPEC] 12.1)
2. Supports Objects ([KMIP-SPEC] 2)
3. Supports Attributes ([KMIP-SPEC] 3)
4. Supports Client-to-Server operations ([KMIP-SPEC] 4)
5. Supports Server-to-Client operations ([KMIP-SPEC] 5)
6. Supports Message Contents ([KMIP-SPEC] 6)
7. Supports Message Formats ([KMIP-SPEC] 7)
8. Supports Authentication ([KMIP-SPEC] 8)
9. Supports Message Encodings ([KMIP-SPEC] 9)
10. Supports Error Handling ([KMIP-SPEC] 11)
11. Optionally supports extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements

5.3 HTTPS Profiles

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.

5.3.1 HTTPS Client

KMIP clients 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. SHOULD support the value */kmip* as the target URI.
4. SHALL enable end user configuration of the target URI used as a KMIP server MAY specify a different target URI.
5. SHALL specify a Content-Type of “application/octet-stream” if the message encoding is TTLV
6. SHALL specify a Content-Type of “text/xml" if the message encoding is XML
7. SHALL specify a Content-Type of “application/json" if the message encoding is JSON
8. SHALL specify a Content-Length
9. SHALL specify a Cache-Control of “no-cache”
10. SHALL send KMIP TTLV message in binary format as the body of the HTTP request

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

5.3.2 HTTPS Server

KMIP servers 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. SHOULD support the value */kmip* as the target URI.
4. SHALL specify a Content-Type of “application/octet-stream” if the message encoding is TTLV
5. SHALL specify a Content-Type of “text/xml" if the message encoding is XML
6. SHALL specify a Content-Type of “application/json" if the message encoding is JSON
7. SHALL specify a Content-Length
8. SHALL specify a Cache-Control of “no-cache”
9. 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 a HTTPS client.

5.3.3 HTTPS Mandatory Test Cases KMIP v1.3

5.3.3.1 MSGENC-HTTPS-M-1-13

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.

See test-cases/kmip-v1.3/mandatory/MSGENC-HTTPS-M-1-13.xml

The informative corresponding wire encoding for the test case is:

```
Request Time 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 03 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-                          ........
```

Response Time 0

```
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 03 00 00 00 00-42 00 92 09 00 00 00 08   ........B.......
00000090: 00 00 00 00 56 8a 5b e2-42 00 0d 02 00 00 00 04   ....V.[bB.......
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.......
```

Request Time 1

```
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 03 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-                          ........
```

Response Time 1

```
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 39 30 34 0d 0a 0d 0a   -Length: 904....
00000050: 42 00 7b 01 00 00 03 80-42 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 03 00 00 00 00-42 00 92 09 00 00 00 08   ........B.......
00000090: 00 00 00 00 56 8a 5b e2-42 00 0d 02 00 00 00 04   ....V.[bB.......
000000a0: 00 00 00 01 00 00 00 00-42 00 0f 01 00 00 03 28   ........B......(
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 03 00-42 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 1d 00 00 00 00-42 00 5c 05 00 00 00 04    ........B.\.....
00000250: 00 00 00 06 00 00 00 00-42 00 5c 05 00 00 00 04    ........B.\.....
00000260: 00 00 00 07 00 00 00 00-42 00 5c 05 00 00 00 04    ........B.\.....
00000270: 00 00 00 1e 00 00 00 00-42 00 5c 05 00 00 00 04    ........B.\.....
00000280: 00 00 00 1b 00 00 00 00-42 00 5c 05 00 00 00 04    ........B.\.....
00000290: 00 00 00 1c 00 00 00 00-42 00 5c 05 00 00 00 04    ........B.\.....
000002a0: 00 00 00 25 00 00 00 00-42 00 5c 05 00 00 00 04    ...%....B.\.....
000002b0: 00 00 00 26 00 00 00 00-42 00 5c 05 00 00 00 04    ...&....B.\.....
000002c0: 00 00 00 1f 00 00 00 00-42 00 5c 05 00 00 00 04    ........B.\.....
000002d0: 00 00 00 20 00 00 00 00-42 00 5c 05 00 00 00 04    ... ....B.\.....
000002e0: 00 00 00 21 00 00 00 00-42 00 5c 05 00 00 00 04    ...!....B.\.....
000002f0: 00 00 00 22 00 00 00 00-42 00 5c 05 00 00 00 04    ..."....B.\.....
00000300: 00 00 00 23 00 00 00 00-42 00 5c 05 00 00 00 04    ...#....B.\.....
00000310: 00 00 00 24 00 00 00 00-42 00 5c 05 00 00 00 04    ...$....B.\.....
00000320: 00 00 00 27 00 00 00 00-42 00 5c 05 00 00 00 04    ...'....B.\.....
00000330: 00 00 00 28 00 00 00 00-42 00 5c 05 00 00 00 04    ...(....B.\.....
00000340: 00 00 00 29 00 00 00 00-42 00 57 05 00 00 00 04    ...)....B.W.....
00000350: 00 00 00 01 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
00000360: 00 00 00 02 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
00000370: 00 00 00 07 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
00000380: 00 00 00 03 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
00000390: 00 00 00 04 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
000003a0: 00 00 00 06 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
000003b0: 00 00 00 08 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
000003c0: 00 00 00 05 00 00 00 00-42 00 57 05 00 00 00 04    ........B.W.....
000003d0: 00 00 00 09 00 00 00 00-                           ........
```

5.4 XML Profiles

The XML profile specifies the use of KMIP replacing the TTLV message encoding with an XML message encoding. The results returned using the XML encoding SHALL be logically the same as if the message encoding was in TTLV form. All size or length values specified within tag values for KMIP items SHALL be the same in XML form as if the message encoding were in TTLV form. The implications of this are that items such as MaximumResponseSize are interpreted to refer to a maximum length computed as if it were a TTLV-encoded response, not the length of the JSON-encoded response.

5.4.1 XML Encoding

5.4.1.1 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:

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

5.4.1.2 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.

5.4.1.3 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.

5.4.1.4 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.

5.4.1.5 Value

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

5.4.1.6 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

5.4.1.6.1 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.

```
<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"/>
```

5.4.1.6.2 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.

5.4.1.6.3 Integer

For XML, value is a decimal and uses [XML-SCHEMA] type xsd:int

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

5.4.1.6.4 Integer - Special case for Masks

(Cryptographic Usage Mask, Storage Status Mask):

Integer mask values can also be encoded as a String containing mask components. XML uses an attribute with [XML-SCHEMA] 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"/>
```

5.4.1.6.5 Long Integer

For XML, value uses [XML-SCHEMA] type xsd:long

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

5.4.1.6.6 Big Integer

For XML, value uses [XML-SCHEMA] type xsd:hexBinary

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

5.4.1.6.7 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"/>
```

5.4.1.6.8 Boolean

For XML, value uses [XML-SCHEMA] type xsd:Boolean

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

5.4.1.6.9 Text String

XML uses [XML-SCHEMA] type xsd:string

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

5.4.1.6.10 Byte String

XML uses [XML-SCHEMA] type xsd:hexBinary

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

5.4.1.6.11 Date-Time

For XML, value uses [XML-SCHEMA] type xsd:dateTime

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

5.4.1.6.12 Interval

XML uses [XML-SCHEMA] type xsd:unsignedInt

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

5.4.2 XML Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)

2. SHALL conform with XML Encoding (5.4.1)
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.5.2
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.4.3 XML Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL conform with XML Encoding (5.4.1)
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.5.3
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.4.4 XML Mandatory Test Cases KMIP v1.3

5.4.4.1 MSGENC-XML-M-1-13

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.

See test-cases/kmip-v1.3/mandatory/MSGENC-XML-M-1-13.xml

5.5 JSON Profiles

The JSON profile specifies the use of KMIP replacing the TTLV message encoding with a JSON message encoding. The results returned using the JSON encoding SHALL be logically the same as if the message encoding was in TTLV form. All size or length values specified within tag values for KMIP items SHALL be the same in JSON form as if the message encoding were in TTLV form. The implications of this are that items such as MaximumResponseSize are interpreted to refer to a maximum length computed as if it were a TTLV-encoded response, not the length of the JSON-encoded response.

5.5.1 JSON Encoding

5.5.1.1 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:

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

5.5.1.2 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.

5.5.1.3 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.

5.5.1.4 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.

5.5.1.5 Value

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

5.5.1.6 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.

5.5.1.6.1 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"}
```

5.5.1.6.2 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.

5.5.1.6.3 Integer

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

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

5.5.1.6.4 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"}
```

5.5.1.6.5 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.

5.5.1.6.6 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"}
```

5.5.1.6.7 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"}
```

5.5.1.6.8 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"}
```

5.5.1.6.9 Text String

For JSON, value must be a String

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

5.5.1.6.10 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"}
```

5.5.1.6.11 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"}
```

5.5.1.6.12 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"}
```

5.5.2 JSON Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL conform with JSON Encoding (5.5.1)
3. SHALL support encoding
4. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.5.2
5. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.5.3 JSON Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL conform with JSON Encoding (5.5.1)
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.5.3
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.5.4 JSON Mandatory Test Cases KMIP v1.3

5.5.4.1 MSGENC-JSON-M-1-13

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.

See test-cases/kmip-v1.3/mandatory/MSGENC-JSON-M-1-13.xml

The informative corresponding wire encoding for the test case is:

```
Request Time 0
{"tag":"RequestMessage", "value":[
  {"tag":"RequestHeader", "value":[
    {"tag":"ProtocolVersion", "value":[
      {"tag":"ProtocolVersionMajor", "type":"Integer",
"value":"0x00000001"},
      {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000003"}
    ]},
    {"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"}
    ]}
  ]}
]}
```

```
Response Time 0
{"tag":"ResponseMessage", "value":[
  {"tag":"ResponseHeader", "value":[
    {"tag":"ProtocolVersion", "value":[
      {"tag":"ProtocolVersionMajor", "type":"Integer",
"value":"0x00000001"},
      {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000003"}
    ]},
    {"tag":"TimeStamp", "type":"DateTime", "value":"2016-01-
04T11:47:46+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"}
  ]}
]}
```

Request Time 1

```
{"tag":"RequestMessage", "value":[
  {"tag":"RequestHeader", "value":[
    {"tag":"ProtocolVersion", "value":[
      {"tag":"ProtocolVersionMajor", "type":"Integer",
"value":"0x00000001"},
      {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000003"}
    ]},
    {"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"}
    ]}
  ]}
]}
```

Response Time 1

```
{"tag":"ResponseMessage", "value":[
  {"tag":"ResponseHeader", "value":[
    {"tag":"ProtocolVersion", "value":[
      {"tag":"ProtocolVersionMajor", "type":"Integer",
"value":"0x00000001"},
      {"tag":"ProtocolVersionMinor", "type":"Integer", "value":"0x00000003"}
    ]},
    {"tag":"TimeStamp", "type":"DateTime", "value":"2016-01-
04T11:47:46+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":"ReKeyKeyPair"},
      {"tag":"Operation", "type":"Enumeration", "value":"Certify"},
```



```
      {"tag":"Operation", "type":"Enumeration", "value":"ReCertify"},
      {"tag":"Operation", "type":"Enumeration", "value":"DiscoverVersions"},
      {"tag":"Operation", "type":"Enumeration", "value":"Notify"},
      {"tag":"Operation", "type":"Enumeration", "value":"Put"},
      {"tag":"Operation", "type":"Enumeration", "value":"RNGRetrieve"},
      {"tag":"Operation", "type":"Enumeration", "value":"RNGSeed"},
      {"tag":"Operation", "type":"Enumeration", "value":"Encrypt"},
      {"tag":"Operation", "type":"Enumeration", "value":"Decrypt"},
      {"tag":"Operation", "type":"Enumeration", "value":"Sign"},
      {"tag":"Operation", "type":"Enumeration", "value":"SignatureVerify"},
      {"tag":"Operation", "type":"Enumeration", "value":"MAC"},
      {"tag":"Operation", "type":"Enumeration", "value":"MACVerify"},
      {"tag":"Operation", "type":"Enumeration", "value":"Hash"},
      {"tag":"Operation", "type":"Enumeration", "value":"CreateSplitKey"},
      {"tag":"Operation", "type":"Enumeration", "value":"JoinSplitKey"},
      {"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"},
      {"tag":"ObjectType", "type":"Enumeration", "value":"PGPKey"}
    ]}
  ]}
]}
```

5.6 Symmetric Key Lifecycle Profiles

The Symmetric Key Lifecycle Profile is a KMIP server performing symmetric key lifecycle operations based on requests received from a KMIP client.

5.6.1 Symmetric Key Lifecycle Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.6.1
3. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.6.2 Symmetric Key Lifecycle Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Symmetric Key* [KMIP-SPEC]
    b. *Key Format Type* [KMIP-SPEC]
3. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm* [KMIP-SPEC]
    b. *Object Type* [KMIP-SPEC]
    c. *Process Start Date* [KMIP-SPEC]
    d. *Protect Stop Date* [KMIP-SPEC]
4. SHALL support the following *Client-to-Server* [KMIP-SPEC] operations:

a. *Create* [KMIP-SPEC]

5. SHALL support the following *Message Encoding* [KMIP-SPEC]:
   a. *Cryptographic Algorithm* [KMIP-SPEC] with values:
      i. 3DES
      ii. AES
   b. *Object Type* [KMIP-SPEC] with value:
      iii. Symmetric Key
   c. *Key Format Type* [KMIP-SPEC] with value:
      iv. Raw
      v. Transparent Symmetric Key
6. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.6.2
7. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.6.3 Symmetric Key Lifecycle Mandatory Test Cases KMIP v1.3

5.6.3.1 SKLC-M-1-13

See test-cases/kmip-v1.3/mandatory/SKLC-M-1-13.xml

5.6.3.2 SKLC-M-2-13

See test-cases/kmip-v1.3/mandatory/SKLC-M-2-13.xml

5.6.3.3 SKLC-M-3-13

See test-cases/kmip-v1.3/mandatory/SKLC-M-3-13.xml

5.6.4 Symmetric Key Lifecycle Optional Test Cases KMIP v1.3

5.6.4.1 SKLC-O-1-13

See test-cases/kmip-v1.3/optional/SKLC-O-1-13.xml

5.7 Symmetric Key Foundry for FIPS 140 Profiles

The Symmetric Key Lifecycle Profile is a KMIP server performing symmetric key lifecycle operations based on requests received from a KMIP client. The use of algorithms within this profile set has been limited to those permitted under the NIST FIPS 140 validation program.

5.7.1 Basic Symmetric Key Foundry Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.7.1
3. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.7.2 Intermediate Symmetric Key Foundry Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.7.2
3. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.7.3 Advanced Symmetric Key Foundry Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.7.3
3. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.7.4 Symmetric Key Foundry Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Symmetric Key [KMIP-SPEC]*
    b. *Key Format Type [KMIP-SPEC]*
3. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm* [KMIP-SPEC]
    b. *Cryptographic Length [KMIP-SPEC] with values:*
        i. *168 (3DES)*
        ii. *128 (AES)*
        iii. *192 (AES*
        iv. *256 (AES)*
    c. *Object Type* [KMIP-SPEC]
    d. *Process Start Date* [KMIP-SPEC]
    e. *Protect Stop Date* [KMIP-SPEC]
4. SHALL support the following *Client-to-Server* [KMIP-SPEC] operations:
    b. *Create* [KMIP-SPEC]
5. SHALL support the following *Message Encoding* [KMIP-SPEC]:
    d. *Cryptographic Algorithm* [KMIP-SPEC] with values:
        i. 3DES
        ii. AES
    e. *Object Type* [KMIP-SPEC] with value:
        i. Symmetric Key
    f. *Key Format Type* [KMIP-SPEC] with value:
        i. Raw
        ii. Transparent Symmetric Key
6. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.7.4

7. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.7.5 Basic Symmetric Key Foundry Mandatory Test Cases KMIP v1.3

5.7.5.1 SKFF-M-1-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-1-13.xml

5.7.5.2 SKFF-M-2-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-2-13.xml

5.7.5.3 SKFF-M-3-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-3-13.xml

5.7.5.4 SKFF-M-4-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-4-13.xml

5.7.6 Intermediate Symmetric Key Foundry Mandatory Test Cases KMIP v1.3

5.7.6.1 SKFF-M-5-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-5-13.xml

5.7.6.2 SKFF-M-6-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-6-13.xml

5.7.6.3 SKFF-M-7-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-7-13.xml

5.7.6.4 SKFF-M-8-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-8-13.xml

5.7.7 Advanced Symmetric Key Foundry Mandatory Test Cases KMIP v1.3

5.7.7.1 SKFF-M-9-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-9-13.xml

5.7.7.2 SKFF-M-10-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-10-13.xml

5.7.7.3 SKFF-M-11-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-11-13.xml

5.7.7.4 SKFF-M-12-13

See test-cases/kmip-v1.3/mandatory/SKFF-M-12-13.xml

5.8 Asymmetric Key Lifecycle Profiles

The Asymmetric Key Lifecycle Profile is a KMIP server performing symmetric key lifecycle operations based on requests received from a KMIP client.

5.8.1 Asymmetric Key Lifecycle Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.8.1
3. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.8.2 Asymmetric Key Lifecycle Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Symmetric Key* [KMIP-SPEC]
    b. *Key Format Type* [KMIP-SPEC]
1. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Public Key* [KMIP-SPEC]
    b. *Private Key* [KMIP-SPEC]
    c. *Key Format Type* [KMIP-SPEC]
2. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm* [KMIP-SPEC]
    b. *Object Type* [KMIP-SPEC]
    c. *Process Start Date* [KMIP-SPEC]
    d. *Process Stop Date* [KMIP-SPEC]
3. SHALL support the following *Message Encoding* [KMIP-SPEC]:
    a. *Cryptographic Algorithm* [KMIP-SPEC] with values:
        i. RSA
    b. *Object Type* [KMIP-SPEC] with value:
        i. Public Key
        ii. Private Key
    c. *Key Format Type* [KMIP-SPEC] with value:
        i. PKCS#1
        ii. PKCS#8
        iii. Transparent RSA Public Key
        iv. Transparent RSA Private Key
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.8.2
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.8.3 Asymmetric Key Lifecycle Mandatory Test Cases KMIP v1.3

5.8.3.1 AKLC-M-1-13

See test-cases/kmip-v1.3/mandatory/AKLC-M-1-13.xml

5.8.3.2 AKLC-M-2-13

See test-cases/kmip-v1.3/mandatory/AKLC-M-2-13.xml

5.8.3.3 AKLC-M-3-13

See test-cases/kmip-v1.3/mandatory/AKLC-M-3-13.xml

5.8.4 Asymmetric Key Lifecycle Optional Test Cases KMIP v1.3

5.8.4.1 AKLC-O-1-13

See test-cases/kmip-v1.3/optional/AKLC-O-1-13.xml

5.9 Cryptographic Profiles

The Basic Cryptographic Client and Server profiles specify the use of KMIP to request encryption and decryption operations from a KMIP server.

The Advanced Cryptographic Client and Server profiles specify the use of KMIP to request encryption, decryption, signature, and verification operations from a KMIP server.

The RNG Cryptographic Client and Server profiles specify the use of KMIP to request random number generator operations from a KMIP server.

5.9.1 Basic Cryptographic Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL support at least one of the *Client-to-Server Operation* [KMIP-SPEC]:
    a. *Encrypt* [KMIP-SPEC]
    b. *Decrypt* [KMIP-SPEC]
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.9.1
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.9.2 Advanced Cryptographic Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL support at least one of the *Client-to-Server Operation* [KMIP-SPEC]:
    a. *Encrypt* [KMIP-SPEC]
    b. *Decrypt* [KMIP-SPEC]
    c. *Sign* [KMIP-SPEC]
    d. *Signature Verify* [KMIP-SPEC]
    e. *MAC* [KMIP-SPEC]
    f. *MAC Verify* [KMIP-SPEC]

g. *RNG Retrieve* [KMIP-SPEC]
h. *RNG Seed* [KMIP-SPEC]

3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.9.2
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.9.3 RNG Cryptographic Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL support at least one of the *Client-to-Server Operation* [KMIP-SPEC]:
   a. *RNG Retrieve* [KMIP-SPEC]
   b. *RNG Seed* [KMIP-SPEC]
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.9.3
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.9.4 Basic Cryptographic Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Client-to-Server Operation* [KMIP-SPEC]:
   a. *Encrypt* [KMIP-SPEC]
   b. *Decrypt* [KMIP-SPEC]
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.9.4
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.9.5 Advanced Cryptographic Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Client-to-Server Operation* [KMIP-SPEC]:
   a. *Encrypt* [KMIP-SPEC]
   b. *Decrypt* [KMIP-SPEC]
   c. *Sign* [KMIP-SPEC]
   d. *Signature Verify* [KMIP-SPEC]
   e. *MAC* [KMIP-SPEC]
   f. *MAC Verify* [KMIP-SPEC]
   g. *RNG Retrieve* [KMIP-SPEC]
   h. *RNG Seed* [KMIP-SPEC]
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.9.5
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.9.6 RNG Cryptographic Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Client-to-Server Operation* [KMIP-SPEC]:
    a. *RNG Retrieve* [KMIP-SPEC]
    b. *RNG Seed* [KMIP-SPEC]
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.9.6
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.9.7 Basic Cryptographic Mandatory Test Cases KMIP v1.3

5.9.7.1 CS-BC-M-1-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-1-13.xml

5.9.7.2 CS-BC-M-2-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-2-13.xml

5.9.7.3 CS-BC-M-3-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-3-13.xml

5.9.7.4 CS-BC-M-4-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-4-13.xml

5.9.7.5 CS-BC-M-5-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-5-13.xml

5.9.7.6 CS-BC-M-6-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-6-13.xml

5.9.7.7 CS-BC-M-7-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-7-13.xml

5.9.7.8 CS-BC-M-8-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-8-13.xml

5.9.7.9 CS-BC-M-9-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-9-13.xml

5.9.7.10 CS-BC-M-10-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-10-13.xml

5.9.7.11 CS-BC-M-11-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-11-13.xml

5.9.7.12 CS-BC-M-12-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-12-13.xml

5.9.7.13 CS-BC-M-13-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-13-13.xml

5.9.7.14 CS-BC-M-14-13

See test-cases/kmip-v1.3/mandatory/CS-BC-M-14-13.xml

5.9.8 Advanced Cryptographic Mandatory Test Cases KMIP v1.3

5.9.8.1 CS-AC-M-1-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-1-13.xml

5.9.8.2 CS-AC-M-2-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-2-13.xml

5.9.8.3 CS-AC-M-3-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-3-13.xml

5.9.8.4 CS-AC-M-4-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-4-13.xml

5.9.8.5 CS-AC-M-5-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-5-13.xml

5.9.8.6 CS-AC-M-6-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-6-13.xml

5.9.8.7 CS-AC-M-7-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-7-13.xml

5.9.8.8 CS-AC-M-8-13

See test-cases/kmip-v1.3/mandatory/CS-AC-M-8-13.xml

5.9.9 RNG Cryptographic Mandatory Test Cases KMIP v1.3

5.9.9.1 CS-RNG-M-1-13

See test-cases/kmip-v1.3/mandatory/CS-RNG-M-1-13.xml

5.9.10 RNG Cryptographic Optional Test Cases KMIP v1.3

5.9.10.1 CS-RNG-O-1-13

See test-cases/kmip-v1.3/mandatory/CS-RNG-O-1-13.xml

5.9.10.2 CS-RNG-O-2-13

See test-cases/kmip-v1.3/mandatory/CS-RNG-O-2-13.xml

5.9.10.3 CS-RNG-O-3-13

See test-cases/kmip-v1.3/mandatory/CS-RNG-O-3-13.xml

5.9.10.4 CS-RNG-O-4-13

See test-cases/kmip-v1.3/mandatory/CS-RNG-O-4-13.xml

5.10 Opaque Managed Object Store Profiles

The Opaque Managed Object Store Profile is a KMIP server performing storage related operations on opaque objects based on requests received from a KMIP client.

5.10.1 Opaque Managed Object Store Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.10.1
3. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.10.2 Opaque Managed Object Store Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following Objects [KMIP-SPEC]
    a. *Opaque Object* [KMIP-SPEC]
3. SHALL support the following Attributes [KMIP-SPEC]
    a. *Object Type* [KMIP-SPEC]
4. SHALL support the following Client-to-Server [KMIP-SPEC] operations:
    a. *Register* [KMIP-SPEC]
5. SHALL support the following Message Encoding [KMIP-SPEC]:
    a. *Opaque Data Type* [KMIP-SPEC]
    b. *Object Type* [KMIP-SPEC] with value:
        i. Opaque Object
6. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.10.2
7. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.10.3 Opaque Managed Object Mandatory Test Cases KMIP v1.3

5.10.3.1 OMOS-M-1-13

See test-cases/kmip-v1.3/mandatory/OMOS-M-1-13.xml

5.10.4 Opaque Managed Object Optional Test Cases KMIP v1.3

5.10.4.1 OMOS-O-1-13

See test-cases/kmip-v1.3/optional/OMOS-O-1-13.xml

5.11 Storage Array with Self-Encrypting Drives Profiles

The Storage Array with Self-Encrypting Drives Profile is a storage array containing self-encrypting drives operating as a KMIP client interacting with a KMIP server.

5.11.1 Storage Array with Self-Encrypting Drives Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHOULD NOT use a C*ustom Attribute* [KMIP-SPEC] that duplicates information that is already in standard *Attributes* [KMIP-SPEC]
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.11.1
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.11.2 Storage Array with Self-Encrypting Drives Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Template* [KMIP-SPEC]
    b. *Secret Data* [KMIP-SPEC]
3. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Custom Attribute* [KMIP SPEC]
4. SHALL support the following client-to-server operations:
    a. *Register* [KMIP-SPEC]
5. SHALL support the following *Message Encoding* [KMIP-SPEC]::
    a. *Secret Data Type Enumeration* [KMIP-SPEC] value:
        i. Password
    b. *Object Type Enumeration* [KMIP-SPEC] values:
        i. Secret Data
        ii. Template
    c. *Name Type Enumeration* [KMIP-SPEC] value:
        i. Uninterpreted Text String
6. SHALL support C*ustom Attribute* [KMIP-SPEC] with the following data types and properties:
    a. TextString
7. SHALL support a minimum length of 64 characters for C*ustom Attribute* [KMIP-SPEC] and *Name* [KMIP-SPEC] values where the attribute type is of variable length.
8. SHALL support a minimum of 10 Custom Attribute [KMIP-SPEC] per managed object
9. SHALL support a minimum of 64 characters in Custom Attribute [KMIP-SPEC] names

10. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.11.2
11. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.11.3 Storage Array with Self-Encrypting Drives Mandatory Test Cases KMIP v1.3

5.11.3.1 SASED-M-1-13

Determine server configuration details including operations supported (only the mandatory operations are listed in the response example), objects supported (only the mandatory objects types are listed in the response example), and optional server information.

See test-cases/kmip-v1.3/mandatory/SASED-M-1-13.xml

5.11.3.2 SASED-M-2-13

A template is created and the secret data for the authentication key is then registered. The server must allow the registration of managed objects for Object Groups either by allowed arbitrary values for Object Groups or by pre-configuration of specific Object Groups prior to the storage array registering the authentication key. The authentication key may be a new authentication key or a replacement authentication key.

See test-cases/kmip-v1.3/mandatory/SASED-M-2-13.xml

5.11.3.3 SASED-M-3-13

Locate and retrieve the previously registered authentication key and finally destroy both the authentication key and the template.

See test-cases/kmip-v1.3/mandatory/SASED-M-3-13.xml

5.12 Tape Library Profiles

The Tape Library Profile specifies the behavior of a tape library operating as a KMIP client interacting with a KMIP server.

5.12.1 Tape Library Profiles Terminology

Key Associated Data (KAD)	Part of the tape format. May be segmented into authenticated and unauthenticated fields. KAD usage is detailed in the SCSI SSC-3 standard from the T10 organization available as ANSI INCITS 335-2000.
Hexadecimal Numeric Characters	Case-sensitive, printable, single byte ASCII characters representing the numbers 0 through 9 and uppercase alpha A through F. (US-ASCII characters 30h-39h and 41h-46h). Each byte (single 8-bit numeric value) is represented as two hexadecimal numeric characters with the high-nibble represented by the first (left-most) hexadecimal numeric character and the low-nibble represented by the second (right-most) hexadecimal numeric character.
N(a)	The maximum number of bytes in the tape authenticated KAD field. For LTO4, N(a) is 12 bytes. For LTO5, N(a) is 60 bytes. For LTO6, N(a) is 60 bytes.

N(u)	The maximum number of bytes in the tape unauthenticated KAD field. For LTO4, N(u) is 32 bytes. For LTO5, N(u) is 32 bytes. For LTO6, N(u) is 32 bytes.
N(k)	The maximum number of bytes in the tape format KAD fields – i.e. N(a) + N(u). For LTO4, N(k) is 44 bytes. For LTO5, N(k) is 92 bytes. For LTO6, N(k) is 92 bytes.

5.12.2 Tape Library Application Specific Information

This information applies to Tape Libraries that use the *Application Specific Information* [KMIP-SPEC] attribute to store key identifiers. KMIP clients are not required to use *Application Specific Information* [KMIP-SPEC] however KMIP servers conforming to the Tape Library Profiles are required to support KMIP clients that use *Application Specific Information* [KMIP-SPEC] and KMIP clients that do not use *Application Specific Information* [KMIP-SPEC].

The *Application Specific Information* [KMIP-SPEC] MAY be used to store data that is specific to the application (Tape Library) using the object.

The following Application Namespaces SHOULD be used in the Application Namespace field of the *Application Specific Information* [KMIP-SPEC]:

- LIBRARY-LTO, LIBRARY-LTO4, LIBRARY-LTO5, LIBRARY-LTO6, LIBRARY-LTO7

For backwards compatibility with deployed Tape Library implementations, servers MAY support VENDOR-LIBRARY-LTO as an Application Namespace, where VENDOR is an ASCII string that SHALL NOT be further interpreted and SHALL be handled by the server as if the Application Namespace was set to LIBRARY-LTO.

Application Specific Information [KMIP-SPEC] supports key identifiers being created either on the server or on the client (Tape Library), but not both. This profile specifies use of key identifiers created by the client.

The *Application Specific Information* [KMIP-SPEC] method of key identification relies on the ability to uniquely identify a key based only on its Application Data (preferably), or (alternatively) on some combination of Application Data and *Custom Attributes* [KMIP-SPEC], which the key creator guarantees to be unique within the Application Namespace.

Key identifiers stored in the KMIP server's *Application Specific Information* [KMIP-SPEC] are in ASCII format. Key identifiers stored in the KMIP client's tape format KAD fields are numeric format. The specific algorithm for converting between text and numeric formats is specified below.

All information contained in the tape format's KAD fields is converted to an ASCII string consisting of hexadecimal numeric character pairs as follows:

1. The unauthenticated KAD is converted to text;
2. The authenticated KAD is converted to text and;
3. The converted authenticated KAD text is concatenated to the end of the converted unauthenticated KAD text.

If the implementation uses client-created key identifiers, then the client generates a new identifier in ASCII format that SHALL be unique within the chosen namespace. The source material for generating the string is dependent on client policy. The numeric representation of this identifier SHALL be no larger than the N(k) bytes of the KAD for the tape media being used.

For KMIP clients and servers conforming to this profile, *Application Specific Information* [KMIP-SPEC] SHALL be created by the Tape Library KMIP client based on the tape format's KAD fields as follows:

1. Define an empty output buffer sufficient to contain a string with a maximum length of 2*N(k) bytes.
2. Copy the tape format's unauthenticated KAD (if present) to the output buffer, converting each byte value to exactly two Hexadecimal Numeric Characters. The first byte (i.e., byte 0) of the output buffer is the first byte of unauthenticated KAD.
3. Concatenate the tape format's authenticated KAD to the output buffer, converting each byte value to exactly two Hexadecimal Numeric Characters.

Note: the contents of the unauthenticated KAD and authenticated KAD fields may be less than the maximum permitted lengths; the implementation provides the correct length values to use in the algorithm rather than using fixed maximum length fields.

If *Application Specific Information* [KMIP-SPEC] is supported, then it SHALL be used by the client for locating the object for the purpose of encrypting and decrypting data on tape. The *Application Specific Information* [KMIP-SPEC] value SHALL solely be used for this purpose.

5.12.3 Tape Library Alternative Name

The Tape Library client SHALL assign a text (i.e., human-readable) representation of the media barcode to the *Alternative Name* [KMIP-SPEC] of the object. This SHALL occur on first use of the object for encryption, which normally is when the library requests the server to create the object.

The relationship between key identifiers in *Application Specific Information* [KMIP-SPEC] and *Alternative Name* [KMIP-SPEC] is as follows:

a) The values for both are provided by the client
b) The identifier in *Alternative Name* [KMIP-SPEC] (i.e., the barcode) SHALL be used by the server administrator for finding keys associated with specific tape media (e.g., a server administrator may want to find the key(s) associated with a missing tape cartridge, where the barcode of that tape cartridge is known).
c) The *Alternative Name* [KMIP-SPEC] SHALL NOT be used by a client for locating the object to encrypt or decrypt data, since the value (barcode) is not required to be unique and therefore does not ensure retrieval of the correct key.

5.12.4 Tape Library Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHOULD support *Application Specific Information* [KMIP-SPEC] with Application Data provided by the client in accordance with Tape Library Application Specific Information (5.12.2)
3. SHOULD NOT use a *Custom Attribute* [KMIP-SPEC] that duplicates information that is already in standard *Attributes* [KMIP-SPEC]
4. MAY use x-Barcode as a *Custom Attribute* [KMIP-SPEC] of type Text String to store the barcode
5. SHALL support the following Attributes [KMIP-SPEC]
    a. *Alternative Name* [KMIP-SPEC-1_2]
6. SHALL support the following Message Encoding [KMIP-SPEC-1_2]:
    a. *Alternative Name Type Enumeration* [KMIP-SPEC-1_2] value:
        i. Uninterpreted Text String
7. SHALL store the media barcode information in an Alternative Name [KMIP-SPEC] Attribute [KMIP-SPEC] in accordance with Tape Library Alternative Name (5.12.3)
8. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.12.4

9. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.12.5 Tape Library Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following Objects [KMIP-SPEC]
    a. *Symmetric Key* [KMIP-SPEC]
3. SHALL support the following Attributes [KMIP-SPEC]:
    a. *Name* [KMIP-SPEC]
    b. *Cryptographic Algorithm* [KMIP-SPEC]
    c. *Custom Attribute* [KMIP SPEC]
    d. *Application Specific Information* [KMIP SPEC]
4. SHALL support the following Client-to-Server Operations [KMIP-SPEC]:
    a. *Create* [KMIP-SPEC]
5. SHALL support the following Message Contents [KMIP-SPEC]:
    a. *Batch Order Option* [KMIP-SPEC] value:
        i. True
    b. *Batch Coun*t [KMIP-SPEC] value:
        i. 1 to 32
6. SHALL support the following Message Encoding [KMIP-SPEC]:
    a. *Cryptographic Algorithm Enumeration* [KMIP-SPEC] value:
        i. AES
    b. *Object Type Enumeration* [KMIP-SPEC] value:
        i. Symmetric Key
    c. *Key Format Type Enumeration* [KMIP-SPEC] value:
        i. Raw
    d. *Cryptographic Length* [KMIP-SPEC] value :
        i. 256-bit
    e. *Name Type Enumeration* [KMIP-SPEC] value:
        i. Uninterpreted Text String
7. SHALL support Custom Attribute [KMIP-SPEC] with the following data types and properties:
    a. Text String
    b. Integer
    c. Date Time
8. SHALL support a minimum length of 255 characters for Custom Attribute [KMIP-SPEC] and *Name* [KMIP-SPEC] values where the attribute type is of variable length
9. SHALL support a minimum of 30 Custom Attribute [KMIP-SPEC] per managed object
1. SHALL support a minimum of 64 characters in *Custom Attribute* [KMIP-SPEC] names
2. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Alternative Name* [KMIP-SPEC-1_2]
3. SHALL support the following *Message Encoding* [KMIP-SPEC-1_2]:
    a. *Alternative Name Type Enumeration* [KMIP-SPEC-1_2] value:

i. Uninterpreted Text String

10. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.12.5
11. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.12.6 Tape Library Mandatory Test Cases KMIP v1.3

5.12.6.1 TL-M-1-13

Determine server configuration details including operations supported (only the mandatory operations are listed in the response example), objects supported (only the mandatory objects types are listed in the response example), optional server information, and optional list of application name spaces. Additional information MAY be returned by tape library clients and servers.

See test-cases/kmip-v1.3/mandatory/TL-M-1-13.xml

5.12.6.2 TL-M-2-13

This case may occur when the Write operation starts with the first block on a tape. The implementation may choose which Write operations qualify for creation of a new key. Regardless of the initiating circumstances, the Tape Library requests the server to create a new AES-256 symmetric key with appropriate identifying information which is unique within the Application Namespace.

Additional custom attributes MAY be specified in order to:

- ensure uniqueness of the key identifier when later Locating the key via Application Specific Information

- provide human-readable information (such as the tape Barcode value)

- provide information to support client-specific purposes

Tape Library implementations are not required to use custom attributes and custom attributes within the create request MAY be omitted.

A Tape Library client MAY elect to perform the steps in separate requests. A Tape Library server SHALL support both requests containing multiple batch items and multiple equivalent requests containing single batch items within each request.

See test-cases/kmip-v1.3/mandatory/TL-M-2-13.xml

5.12.6.3 TL-M-3-13

The Tape Library constructs an identifier string based on the method in Tape Library Application Specific Information (5.12.2), and requests the server to locate the matching managed object for that Application Specific Information value. A Get is then requested based on the key's unique identifier. The Tape Library MAY update attributes associated with the Symmetric Key Managed Object. The following test case shows extensive use of custom attributes. Custom attributes are not required if the Application Name is unique within the Application Namespace. An implementation may also use custom attributes for vendor-unique purposes, or to improve usability.

Tape Library implementations are not required to use custom attributes and those steps within the test case that refer to custom attribute setting and update are optional and MAY be omitted. The steps using Get Attribute List, Get Attributes and Modify Attribute are optional for a client to use but remain mandatory for a server to support for those clients that elect to use the custom attributes.

A Tape Library client MAY elect to perform the steps in separate requests. A Tape Library server SHALL support both requests containing multiple batch items and multiple equivalent requests containing single batch items within each request.

The test case destroys the key created in the previous test case to clean up after the test. Tape Library implementations MAY elect to not perform this step.

See test-cases/kmip-v1.3/mandatory/TL-M-3-13.xml

5.13 Suite B Profiles

Suite B [SuiteB] requires that key establishment and signature algorithms be based upon Elliptic Curve Cryptography and that the encryption algorithm be AES [FIPS197]. Suite B includes:

Encryption	Advanced Encryption Standard (AES) (key sizes of 128 and 256 bits)
Digital Signature	Elliptic Curve Digital Signature Algorithm (ECDSA) (using the curves with 256-bit and 384-bit prime moduli)
Key Exchange	Elliptic Curve Diffie-Hellman (ECDH), (using the curves with 256-bit and 384-bit prime moduli)
Hashes	SHA-256 and SHA-384

Suite B provides for two levels of cryptographic security, namely a 128-bit minimum level of security (minLOS_128) and a 192-bit minimum level of security (minLOS_192). Each level defines a minimum strength that all cryptographic algorithms must provide. A KMIP product configured at a minimum level of security of 128 bits provides adequate protection for classified information up to the SECRET level. A KMIP product configured at a minimum level of security of 192 bits is required to protect classified information at the TOP SECRET level.

The Suite B non-signature primitives are divided into two columns as shown below.

	Column 1	**Column 2**
Encryption	AES-128	AES-256
Key Agreement	ECDH on P-256	ECDH on P-384
Hash for PRF/MAC	SHA-256	SHA-384

At the 128-bit minimum level of security, the non-signature primitives MUST either come exclusively from Column 1 or exclusively from Column 2.

At the 192-bit minimum level of security, the non-signature primitives MUST come exclusively from Column 2.

Digital signatures using ECDSA MUST be used for authentication. Following the direction of RFC 4754, ECDSA-256 represents an instantiation of the ECDSA algorithm using the P-256 curve and the SHA-256 hash function. ECDSA-384 represents an instantiation of the ECDSA algorithm using the P-384 curve and the SHA-384 hash function.

If configured at a minimum level of security of 128 bits, a KMIP product MUST use either ECDSA-256 or ECDSA-384 for authentication. It is allowable for one party to authenticate with ECDSA-256 and the other party to authenticate with ECDSA-384. This flexibility will allow interoperability between a KMIP client and server that have different sizes of ECDSA authentication keys. KMIP products configured at a minimum level of security of 128 bits MUST be able to verify ECDSA-256 signatures and SHOULD be able to verify ECDSA-384 signatures. If configured at a minimum level of security of 192 bits, ECDSA-384 MUST be used by both the KMIP client and server for authentication. KMIP products configured at a minimum level of security of 192 bits MUST be able to verify ECDSA-384 signatures.

KMIP products, at both minimum levels of security, MUST each use an X.509 certificate that complies with the "Suite B Certificate and Certificate Revocation List (CRL) Profile" [RFC5759] and that contains an elliptic curve public key with the key usage bit set for digital signature.

5.13.1 Suite B minLOS_128 Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL restrict use of the enumerated types listed in item 6 of Suite B minLOS_128 Server (5.13.2) to the enumeration values noted against each item
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.13.1
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP or [CNSSP-15] requirements.

5.13.2 Suite B minLOS_128 Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Certificate* [KMIP-SPEC]
    b. *Symmetric Key* [KMIP-SPEC]
    c. *Public Key* [KMIP-SPEC]
    d. *Private Key* [KMIP-SPEC]
3. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm* [KMIP-SPEC]
    b. *Cryptographic Length* [KMIP-SPEC] value :
        i. 128-bit (combined with AES)
        ii. 256-bit (combined with SHA, ECDH or ECDSA)
4. MAY support the following *Attributes* [KMIP-SPEC]
    a. *Cryptographic Length* [KMIP-SPEC] value :
        i. 256-bit (combined with AES)
        ii. 384-bit bit (combined with SHA, ECDH or ECDSA)
5. SHALL support the following C*lient-to-Server Operations* [KMIP-SPEC]:
    a. *Create* [KMIP-SPEC]
    b. *Create Key Pair* [KMIP-SPEC]
    c. *Register* [KMIP-SPEC]
    d. *Re-key* [KMIP-SPEC]
    e. *Re-key Key Pair* [KMIP-SPEC]
6. SHALL support the following *Message Encoding* [KMIP-SPEC]:
    a. *Recommended Curve Enumeration* [KMIP-SPEC] value:
        i. P-256 (SECP256R1)
    b. *Certificate Type Enumeration* [KMIP-SPEC] value:
        i. X.509
    c. *Cryptographic Algorithm Enumeration* [KMIP-SPEC] value:
        i. AES
        ii. ECDSA
        iii. ECDH
        iv. HMAC-SHA256

d. *Hashing Algorithm Enumeration* [KMIP-SPEC]
    i. SHA-256
e. *Object Type Enumeration* [KMIP-SPEC] value:
    i. Certificate
    ii. Symmetric Key
    iii. Public Key
    iv. Private Key
f. *Key Format Type Enumeration* [KMIP-SPEC] value:
    i. Raw
    ii. ECPrivateKey
    iii. X.509
    iv. Transparent ECDSA Private Key
    v. Transparent ECDSA Public Key
    vi. Transparent ECDH Private Key
    vii. Transparent ECDH Public Key
    viii. Transparent EC Private Key
    ix. Transparent EC Public Key
g. *Digital Signature Algorithm Enumeration* [KMIP-SPEC] value:
    i. ECDSA with SHA256 (on P-256)

7. MAY support the following *Message Encoding* [KMIP-SPEC]:
    a. *Recommended Curve* [KMIP-SPEC] value:
        i. P-384 (SECP384R1)
    b. *Cryptographic Algorithm Enumeration* [KMIP-SPEC] value:
        i. HMAC-SHA384
    c. *Hashing Algorithm Enumeration* [KMIP-SPEC]
        i. SHA-384
    d. Digital Signature Algorithm Enumeration
        i. ECDSA with SHA384 (on P-384)
8. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.13.2
9. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP or [CNSSP-15] requirements.

5.13.3 Suite B minLOS_128 Mandatory Test Cases KMIP v1.3

5.13.3.1 SUITEB_128-M-1-13

Perform a Query operation, querying the Operations and Objects supported by the server, and get a successful response.

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

The TLS protocol version and cipher suite SHALL be as specified in Suite B minLOS_128 Cipher Suites (3.3.2)

See test-cases/kmip-v1.3/mandatory/SUITEB_128-M-1-13.xml

5.13.4 Suite B minLOS_192 Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL restrict use of the enumerated types listed in item 5 of Suite B minLOS_192 Server (5.13.5) to the enumeration values noted against each item
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.13.4
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP or [CNSSP-15] requirements.

5.13.5 Suite B minLOS_192 Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Certificate* [KMIP-SPEC]
    b. *Symmetric Key* [KMIP-SPEC]
    c. *Public Key* [KMIP-SPEC]
    d. *Private Key* [KMIP-SPEC]
3. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm* [KMIP-SPEC]
    b. *Cryptographic Length* [KMIP-SPEC] value :
        i. 256-bit (combined with AES)
        ii. 384-bit (combined with SHA, ECDH or ECDSA)
4. SHALL support the following C*lient-to-Server Operations* [KMIP-SPEC]:
    a. *Create* [KMIP-SPEC]
    b. *Create Key Pair* [KMIP-SPEC]
    c. *Register* [KMIP-SPEC]
    d. *Re-key* [KMIP-SPEC]
    e. *Re-key Key Pair* [KMIP-SPEC]
5. SHALL support the following *Message Encoding* [KMIP-SPEC]:
    a. *Recommended Curve Enumeration* [KMIP-SPEC] value:
        i. P-384 (SECP384R1)
    b. *Certificate Type Enumeration* [KMIP-SPEC] value:
        i. X.509
    c. *Cryptographic Algorithm Enumeration* [KMIP-SPEC] value:
        i. AES
        ii. ECDSA
        iii. ECDH
        iv. HMAC-SHA384
    d. *Hashing Algorithm Enumeration* [KMIP-SPEC]
        i. SHA-384
    e. *Object Type Enumeration* [KMIP-SPEC] value:
        i. Certificate

ii. Symmetric Key
iii. Public Key
iv. Private Key

f. *Key Format Type Enumeration* [KMIP-SPEC] value:
i. Raw
ii. ECPrivateKey
iii. X.509
iv. Transparent ECDSA Private Key
v. Transparent ECDSA Public Key
vi. Transparent ECDH Private Key
vii. Transparent ECDH Public Key
viii. Transparent EC Private Key
ix. Transparent EC Public Key

g. *Digital Signature Algorithm Enumeration* [KMIP-SPEC] value:
i. ECDSA with SHA384 (on P-384)

6. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.13.5
7. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP or [CNSSP-15] requirements.

5.13.6 Suite B minLOS_192 Mandatory Test Cases KMIP v1.3

5.13.6.1 SUITEB_192-M-1-13

Perform a Query operation, querying the Operations and Objects supported by the server, and get a successful response.

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

The TLS protocol version and cipher suite SHALL be as specified in Suite B minLOS_192 Cipher Suites (3.4.2)

See test-cases/kmip-v1.3/mandatory/SUITEB_192-M-1-13.xml

6 Conformance

The baseline server and client profiles provide the most basic functionality that is expected of a conformant KMIP client or server. The complete server profile defines a KMIP server that implements the entire specification. A KMIP implementation conformant to this specification (the Key Management Interoperability Protocol Profiles) SHALL meet all the conditions documented in one or more of the following sections.

6.1 Baseline Client Basic KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Baseline Client conditions (5.1.1)

6.2 Baseline Client TLS v1.2 KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the TLS 1.2 Authentication Suite conditions (3.2) and;
3. SHALL support the Baseline Client conditions (5.1.1)

6.3 Baseline Server Basic KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Baseline Server conditions (5.1.2)

6.4 Baseline Server TLS v1.2 KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the TLS 1.2 Authentication Suite conditions (3.2) and;
3. SHALL support the Baseline Server conditions (5.1.2)

6.5 Complete Server Basic KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Complete Server conditions (5.2)

6.6 Complete Server TLS v1.2 KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the TLS 1.2 Authentication Suite conditions (3.2) and;

3. SHALL support the Complete Server conditions (5.2)

6.7 HTTPS Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the HTTPS Authentication Suite conditions (3.5) and;
3. SHALL support the HTTPS Client conditions (5.3.1) and;
4. SHALL support all of the HTTPS Mandatory Test Cases KMIP v1.3 (5.3.3).

6.8 HTTPS Server KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the HTTPS Authentication Suite conditions (3.5) and;
3. SHALL support the HTTPS Server conditions (5.3.2) and;
4. SHALL support all of the HTTPS Mandatory Test Cases KMIP v1.3 (5.3.3).

6.9 XML Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the XML Client conditions (5.4.2) and;
4. SHALL support one or more of the XML Mandatory Test Cases KMIP v1.3 (5.4.4).

6.10 XML Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the XML Server conditions (5.4.3) and;
4. SHALL support mapping to/from XML of all TTLV tags and enumerations specified within [KMIP-SPEC] and;
5. SHALL support all of the XML Mandatory Test Cases KMIP v1.3 (5.4.4).

6.11 JSON Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the JSON Client conditions (5.5.2) and;
4. SHALL support one or more of the JSON Mandatory Test Cases KMIP v1.3 (5.5.4).

6.12 JSON Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;

3. SHALL support the JSON Client conditions (5.5.2) and;
4. SHALL support mapping to/from JSON all TTLV tags and enumerations specified within [KMIP-SPEC] and;
5. SHALL support all of the JSON Mandatory Test Cases KMIP v1.3 (5.5.4).

6.13 Symmetric Key Lifecycle Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Symmetric Key Lifecycle Client conditions (5.6.1) and;
4. SHALL support one or more of the Symmetric Key Lifecycle Mandatory Test Cases KMIP v1.3 (5.6.3).

6.14 Symmetric Key Lifecycle Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Symmetric Key Lifecycle Server conditions (5.6.2) and;
4. SHALL support all of the Symmetric Key Lifecycle Mandatory Test Cases KMIP v1.3 (5.6.3).

6.15 Basic Symmetric Key Foundry Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Basic Symmetric Key Foundry Client conditions (5.7.1) and;
4. SHALL support one or more of the Basic Symmetric Key Foundry Mandatory Test Cases KMIP v1.3 (5.7.5).

6.16 Intermediate Symmetric Key Foundry Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Basic Symmetric Key Foundry Client conditions (5.7.1) and;
4. SHALL support one or more of the Intermediate Symmetric Key Foundry Mandatory Test Cases KMIP v1.3 (5.7.6).

6.17 Advanced Symmetric Key Foundry Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Basic Symmetric Key Foundry Client conditions (5.7.1) and;

4. SHALL support one or more of the Advanced Symmetric Key Foundry Mandatory Test Cases KMIP v1.3 (5.7.7).

6.18 Symmetric Key Foundry Server KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Symmetric Key Foundry Server conditions (5.7.4) and;
4. SHALL support all of the Basic Symmetric Key Foundry Mandatory Test Cases KMIP v1.3 (5.7.5).
5. SHALL support all of the Intermediate Symmetric Key Foundry Mandatory Test Cases KMIP v1.3 (5.7.6).
6. SHALL support all of the Advanced Symmetric Key Foundry Mandatory Test Cases KMIP v1.3 (5.7.7).

6.19 Asymmetric Key Lifecycle Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Asymmetric Key Lifecycle Client conditions (5.6.1) and;
4. SHALL support one or more of the Asymmetric Key Lifecycle Mandatory Test Cases KMIP v1.3 (5.8.3).

6.20 Asymmetric Key Lifecycle Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Asymmetric Key Lifecycle Client conditions (5.6.2) and;
4. SHALL support all of the Asymmetric Key Lifecycle Mandatory Test Cases KMIP v1.3 (5.8.3).

6.21 Basic Cryptographic Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Basic Cryptographic Client conditions (5.9.1) and;
5. SHALL support one or more of the Basic Cryptographic Mandatory Test Cases KMIP v1.3 (5.9.7).

6.22 Advanced Cryptographic Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Advanced Cryptographic Client conditions (5.9.2) and;
4. SHALL support one or more of the Advanced Cryptographic Mandatory Test Cases KMIP v1.3 (5.9.8).

6.23 RNG Cryptographic Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the RNG Cryptographic Client conditions (5.9.3) and;
4. SHALL support one or more of the RNG Cryptographic Mandatory Test Cases KMIP v1.3 (5.9.9).

6.24 Basic Cryptographic Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Basic Cryptographic Server conditions (5.9.4) and;
6. SHALL support all of the Basic Cryptographic Mandatory Test Cases KMIP v1.3 (5.9.7).

6.25 Advanced Cryptographic Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Advanced Cryptographic Server conditions (5.9.5) and;
4. SHALL support all of the Advanced Cryptographic Mandatory Test Cases KMIP v1.3 (5.9.8).

6.26 RNG Cryptographic Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the RNG Cryptographic Server conditions (5.9.6) and;
4. SHALL support all of the RNG Cryptographic Mandatory Test Cases KMIP v1.3 (5.9.9).

6.27 Opaque Managed Object Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Opaque Managed Object Store Client conditions (5.10.1) and;
4. SHALL support one or more of the Opaque Managed Object Mandatory Test Cases KMIP v1.3 (5.10.3).

6.28 Opaque Managed Object Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Opaque Managed Object Store Server conditions (5.10.2) and;
4. SHALL support all of the Opaque Managed Object Mandatory Test Cases KMIP v1.3 (5.10.3).

6.29 Storage Array with Self-Encrypting Drives Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Storage Array with Self-Encrypting Drives Client conditions (5.11.1) and;
4. SHALL support one or more of the Storage Array with Self-Encrypting Drives Mandatory Test Cases KMIP v1.3 (5.11.3).

6.30 Storage Array with Self-Encrypting Drives Server KMIP v1.3 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Storage Array with Self-Encrypting Drives Server conditions (5.11.2) and;
4. SHALL support all of the Storage Array with Self-Encrypting Drives Mandatory Test Cases KMIP v1.3 (5.11.3).

6.31 Tape Library Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Tape Library Client conditions (5.12.4) and;
4. SHALL support the Tape Library Application Specific Information conditions (5.12.2) and;
5. SHALL support the Tape Library Alternative Name conditions (5.12.3) and;
6. SHALL support one or more of the Tape Library Mandatory Test Cases KMIP v1.3 (5.12.6).

6.32 Tape Library Server KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Basic Authentication Suite conditions (3.1) and;
3. SHALL support the Tape Library Server conditions (5.12.5) and;
4. SHALL support the Tape Library Application Specific Information conditions (5.12.2) and;
5. SHALL support the Tape Library Alternative Name conditions (5.12.3) and;
6. SHALL support all of the Tape Library Mandatory Test Cases KMIP v1.3 (5.12.6).

6.33 Suite B minLOS_128 Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Suite B minLOS_128 Authentication Suite (3.3) and;
3. SHALL support the Suite B minLOS_128 Client conditions (5.13.1) and;
4. SHALL support one or more of the Suite B minLOS_128 Mandatory Test Cases KMIP v1.3 (5.13.3).

6.34 Suite B minLOS_128 Server KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Suite B minLOS_128 Authentication Suite (3.3) and;
3. SHALL support the Suite B minLOS_128 Server conditions (5.13.2) and;
4. SHALL support all of the Suite B minLOS_128 Mandatory Test Cases KMIP v1.3 (5.13.3).

6.35 Suite B minLOS_192 Client KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Suite B minLOS_192 Authentication Suite (3.4) and;
3. SHALL support the Suite B minLOS_192 Client conditions (5.13.4) and;
4. SHALL support one or more of the Suite B minLOS_192 Mandatory Test Cases KMIP v1.3 (5.13.6).

6.36 Suite B minLOS_192 Server KMIP v1.3 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the Suite B minLOS_192 Authentication Suite (3.4) and;
3. SHALL support the Suite B minLOS_192 Server conditions (5.13.5) and;
4. SHALL support all of the Suite B minLOS_192 Mandatory Test Cases KMIP v1.3 (5.13.6).

Appendix A. Acknowledgments

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

Participants:

Warren Armstrong, QuintessenceLabs Pty Ltd.
Rinkesh Bansal, IBM
Lina Baquero, Fornetix
Jeff Bartell, Fornetix
Tom Benjamin, IBM
Anthony Berglas, Cryptsoft Pty Ltd.
Mathias Bjorkqvist, IBM
Todd Bottger, Oracle
Joseph Brand, Semper Fortis Solutions
Alan Brown, Thales e-Security
Robert Burns, Thales e-Security
Andrew Byrne, EMC
Hai-May Chao, Oracle
Chye-Lin Chee, Hewlett Packard Enterprise (HPE)
Tim Chevalier, NetApp
Kenli Chong, QuintessenceLabs Pty Ltd.
Justin Corlett, Cryptsoft Pty Ltd.
Tony Cox, Cryptsoft Pty Ltd.
DINESH DIALANI, SafeNet, Inc.
Michael Dong, Hewlett Packard Enterprise (HPE)
Alex Downey, Futurex
Kevin Driver, IBM
Stephen Edwards, Fornetix
James Espinoza, Futurex
Faisal Faruqui, Thales e-Security
Stan Feather, Hewlett Packard Enterprise (HPE)
Indra Fitzgerald, NetApp
Judith Furlong, EMC
Michael Gardiner, SafeNet, Inc.
Jonathan Geater, Thales e-Security
Susan Gleeson, Oracle
Saheem Granados, IBM
John Green, QuintessenceLabs Pty Ltd.
Robert Griffin, EMC
Robert Haas, IBM
Steve He, Vormetric, Inc.
Christopher Hiller, Hewlett Packard Enterprise (HPE)
Hao Hoang, Hewlett Packard Enterprise (HPE)
Tim Hudson, Cryptsoft Pty Ltd.
Michael Jenkins, National Security Agency
Elysa Jones, Individual
Mark Joseph, P6R, Inc
Mahadev Karadigudda, NetApp
Jason Katonica, IBM
Tim Kelsey, Hewlett Packard Enterprise (HPE)
Hyun-jin Kim, Hancom Secure, Inc.
Stephen Kingston, SafeNet, Inc.
Kathy Kriese, Symantec Corp.
Leonardo Ladeira, SafeNet, Inc.

Sun-ho Lee, Hancom Secure, Inc.
John Leiseboer, QuintessenceLabs Pty Ltd.
Hal Lockhart, Oracle
Robert Lockhart, Thales e-Security
Martin Luther, Hewlett Packard Enterprise (HPE)
Jane Melia, QuintessenceLabs Pty Ltd.
Prashant Mestri, IBM
Trisha Paine, SafeNet, Inc.
Incheon Park, Hancom Secure, Inc.
John Peck, IBM
Michael Phillips, Dell
Stefan Pingel, EMC
Ajai Puri, SafeNet, Inc.
Saravanan Ramalingam, Thales e-Security
Bruce Rich, Cryptsoft Pty Ltd.
Warren Robbins, Dell
Peter Robinson, EMC
Rick Robinson, IBM
Saikat Saha, Oracle
Boris Schumperli, Cryptomathic
Greg Scott, Cryptsoft Pty Ltd.
Amit Sinha, SafeNet, Inc.
Radhika Siravara, Oracle
Curtis Smith, Futurex
Ryan Smith, Futurex
Amruta Soman, Cryptsoft Pty Ltd.
Gerald Stueve, Fornetix
Jim Susoy, P6R, Inc
Kiran Thota, VMware, Inc.
Peter Tsai, Vormetric, Inc.
Nathan Turajski, Hewlett Packard Enterprise (HPE)
Charles White, Fornetix
Steve Wierenga, Hewlett Packard Enterprise (HPE)
Thomas Xuelin, Watchdata Technologies Pte Ltd.
Krishna Yellepeddy, IBM
Magda, Zdunkiewicz, Cryptsoft Pty Ltd.
yuan zhang, Watchdata Technologies Pte Ltd.
Joshua Zhu, Vormetric, Inc.

Appendix B. Revision History

Revision	Date	Editor	Changes Made
wd02	4-Jan-2016	Tim Hudson	Added in the remainder of the existing KMIP 1.2 profiles updated for KMIP 1.3 incorporating feedback to date on the existing profiles.
wd01	5-Nov-2015	Tim Hudson	Initial revision based on the KMIP 1.2 equivalent documents and TC discussions and TC admin feedback on computer readable files