Key Management Interoperability Protocol Profiles Version 2.0

Committee Specification 01

12 June 2019

This version:
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/cs01/kmip-profiles-v2.0-cs01.docx (Authoritative)
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/cs01/kmip-profiles-v2.0-cs01.html
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/cs01/kmip-profiles-v2.0-cs01.pdf

Previous version:
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/csd01/kmip-profiles-v2.0-csd01.docx(Authoritative)
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/csd01/kmip-profiles-v2.0-csd01.html
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/csd01/kmip-profiles-v2.0-csd01.pdf

Latest version:
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/kmip-profiles-v2.0.docx (Authoritative)
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/kmip-profiles-v2.0.html
https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/kmip-profiles-v2.0.pdf

Technical Committee:
OASIS Key Management Interoperability Protocol (KMIP) TC

Chairs:
Tony Cox (tony.cox@cryptsoft.com), Cryptsoft Pty Ltd.
Judith Furlong (Judith.Furlong@dell.com), Dell

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

Related work:

This specification replaces or supersedes:

- *Key Management Interoperability Protocol Profiles Version 1.4*. Edited by Tim Hudson and Robert Lockhart. 22 November 2017. OASIS Standard. http://docs.oasis-open.org/kmip/profiles/v1.4/os/kmip-profiles-v1.4-os.html. Latest version: http://docs.oasis-open.org/kmip/profiles/v1.4/kmip-profiles-v1.4.html.

This specification is related to:

- *Key Management Interoperability Protocol Specification Version 2.0.* Edited by Tony Cox and Charles White. Latest version: https://docs.oasis-open.org/kmip/kmip-spec/v2.0/kmip-spec-v2.0.html.
- *Key Management Interoperability Protocol Test Cases Version 2.0. Edited by Tim Hudson* and Mark Joseph. Latest version: https://docs.oasis-open.org/kmip/kmip-testcases/v2.0/kmip-testcases-v2.0.html.
- *Key Management Interoperability Protocol Usage Guide Version 2.0.* Edited by Judith Furlong. Latest version: https://docs.oasis-open.org/kmip/kmip-ug/v2.0/kmip-ug-v2.0.html.

Abstract:

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

Status:

This document was last revised or approved by the 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/.

This specification is provided under the RF on RAND Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. 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).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

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

[kmip-profiles-v2.0]

Key Management Interoperability Protocol Profiles Version 2.0. Edited by Tim Hudson and Robert Lockhart. 12 June 2019. OASIS Committee Specification 01. https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/cs01/kmip-profiles-v2.0-cs01.html. Latest version: https://docs.oasis-open.org/kmip/kmip-profiles/v2.0/kmip-profiles-v2.0.html.

Notices

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

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

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

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

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

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

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

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see 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 2.2.6 for the KMIP Specification [KMIP-SPEC] 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 IPR Policy

This specification is provided under the RF on RAND Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. 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).

1.2 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].

1.3 Normative References

[KMIP-SPEC] *Key Management Interoperability Protocol Specification Version 2.0.* Edited by Tony Cox and Charles White. Latest version: https://docs.oasis-open.org/kmip/kmip-spec/v2.0/kmip-spec-v2.0.html.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. https://www.ietf.org/rfc/rfc2119.txt.

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

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

[RFC8446] E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.3*, IETF RFC 8446, August 2018, https://www.ietf.org/rfc/rfc8446.txt.

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

[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/

1.4 Non-Normative References

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

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

[TC-PROC] *OASIS TC Process.* 1 July 2017. 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 if versions other than [KMIP-SPEC] are 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 *Object Attributes* that SHALL be supported
5. Specify the list of *Operations* that SHALL be supported
6. Specify any other requirements that SHALL be supported
7. Specify the mandatory test cases that SHALL be supported by conforming implementations
8. 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 final 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 SHALL provide assurance of client authenticity
4. All options such as channel protocol version and cipher suites for the secuity channel SHALL be specified

When using automated client provisioning, the assurance of server authenticity and client authenticity MAY be provided via means outside of the security channel protocol.

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 servers SHALL support:

- TLS v1.3 [RFC8446]

Conformant KMIP clients SHOULD support:

- TLS v1.3 [RFC8446]

Conformant KMIP servers SHOULD support:

- TLS v1.2 [RFC5246]

Conformant KMIP clients MAY support:

- TLS v1.2 [RFC5246]

Conformant KMIP clients or servers SHALL NOT support:

- TLS v1.1 [RFC4346]
- TLS v1.0 [RFC2246]
- Any version of the SSL protocol

3.1.2 Basic Authentication Cipher Suites

Conformant KMIP servers SHALL support all of the following cipher suites for TLSv1.3 if TLSv1.3 is supported:

- TLS13-CHACHA20-POLY1305-SHA256
- TLS13-AES-256-GCM-SHA384

Conformant KMIP clients SHALL support at least one of the following cipher suites for TLSv1.3 if TLSv1.3 is supported:

- TLS13-CHACHA20-POLY1305-SHA256
- TLS13-AES-256-GCM-SHA384

Conformant KMIP clients or servers SHALL support the following cipher suites for TLSv1.2 if TLSv1.2 is supported:

- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256

Conformant KMIP clients or servers MAY support the following cipher suites for TLSv1.2 if TLSv1.2 is supported:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_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_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
- 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_AES_128_CBC_SHA
- TLS_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.

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 except when using automated client provisioning.

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 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 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.

3.2.1 HTTPS Protocols

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

3.2.2 HTTPS Cipher Suites

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

3.2.3 HTTPS Authenticity

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

3.2.4 HTTPS KMIP Port Number

KMIP servers conformant to this profile SHOULD 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. Unique Identifier
2. Private Key Unique Identifier
3. Public Key Unique Identifier
4. Unique Batch Item ID
5. Asynchronous Correlation Value
6. Time Stamp
7. Key Value / Key Material 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 / Signature Data/ MAC Data / IV Counter Nonce 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. Activation Date
    b. Archive Date

    c. Compromise Date
    d. Compromise Occurrence Date
    e. Deactivation Date
    f. Destroy Date
    g. Initial Date
    h. Last Change Date
    i. Protect Start Date
    j. Process Stop Date
    k. Validity Date
    l. Original Creation Date
10. Linked Object Identifier
11. Digest Value
    a. For those managed cryptographic objects which are dynamically generated
12. Key Format Type
    a. The key format type selected by the server when it creates managed objects except when the key format type is specified in the request or there is a default value required in the specification and in which case the value must match.
13. Digest
    a. The Hashing Algorithm selected by the server when it calculates the digest for a managed object for which it has access to the key material provided that a Digest is always available with the Hashing Algorithm of SHA256 (the default)
    b. The Digest Value
14. Extensions reported in Query for function Query Extension List and Query Extension Map
15. Application Namespaces reported in Query for function Query Application Namespaces
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. Server Correlation Value
20. Client Correlation Value
21. 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. Vendor Attributes – 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. Result Message – May or may not be included in responses and the value (if included) may vary from the text contained within the test case.
5. The list of Protocol Versions returned in a Discover Version response may include additional protocol versions if the request has not specified a list of client supported Protocol Versions.
6. Vendor Identification - The value (if included) may vary from the text contained within the test case.

7. Random Number Generator – The value returned may vary from the value returned including any of the defined values for the RNG Algorithm field within the Random Number Generator attribute including Unspecified. The other fields within the Random Number Generator (all of which are defined as optional) may be present or omitted and their value each field may be set to any value that is permitted for such a field.
8. Located Items – The field MAY be present in responses to Locate even if an Offset Items field is not present in the request.
9. Server Information – the contents of the structure returned MAY vary although the structure itself SHALL be present if specified in the response.

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* (where the name includes the test identifier); and
    b. *InteropIdentifier*
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 logically equivalent
4. A request MAY contain an optional *Authentication* [KMIP_SPEC] structure within each request
5. The order of Attributes returned in a *Get Attributes* 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 *Batch Item* of a client request including multiple batch items (i.e. Batch Count is greater than one) the *Unique Batch Item ID* must be specified. *Unique Batch Item ID* MAY be specified for client requests containing a single batch item (*Batch Count* equals one) or MAY be omitted. Any reasonable appearing datum of the expected type is permitted.
7. A test MAY be preceded by a request containing the Operation *Interop* with the Interop Function set to *Begin*
8. A test MAY be followed by a request containing the Operation *Interop* with the Interop Function set to *End*
9. Use of the Operation *Interop* is optional (it is not expected that production KMIP clients will support the *Interop* Operation) however the use of the *Interop* function during a formal interop test event may be mandatory (depending on the rules of the specific interop event).

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 if it meets the following conditions:

1. Supports the conditions required by the *KMIP Client Implementation Conformance* clauses [KMIP-SPEC]
2. Supports the following *Attribute Data Structures* [KMIP-SPEC]:
    a. Attributes
3. Supports the following *Object Attributes* [KMIP-SPEC]:
    a. *Activation Date*
    b. *Deactivation Date*
    c. *Digest*
    d. *Initial Date*
    e. *Last Change Date*
    f. *Object Type*
    g. *State*
    h. *Unique Identifier*
4. Supports the following *Client-to-Server Operations* [KMIP-SPEC]:
    a. *Get*
    b. *Get Attributes*
    c. *Locate*
    d. *Query*
5. Supports the following *Message Data Structures* [KMIP-SPEC]:
    a. *Asynchronous Indicator*
    b. *Batch Count*
    c. *Batch Error Continuation Option*
    d. *Batch Item*
    e. *Batch Order Option*
    f. *Maximum Response Size*
    g. *Message Extension*
    h. *Operation*
    i. *Protocol Version*
    j. *Result Reason*
    k. *Result Status*
    l. *Server Correlation Value*
    m. *Time Stamp*
    n. *Unique Batch Item ID*
6. Supports the *Message Protocols* [KMIP-SPEC]
    a. *Authentication*
    b. *Transport*
    c. *TTLV*
7. Optionally supports any clause within [KMIP-SPEC] that is not listed above.
8. 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.

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

1. Supports the conditions required by the *KMIP Server Implementation Conformance* clauses [KMIP-SPEC]
2. Supports the following *Attribute Data Structures* [KMIP-SPEC]:
    a. *Attributes*
3. Supports the following *Message Data Structures* [KMIP-SPEC]:
    a. *Credential*
4. Supports the following *Object Data Structures* [KMIP-SPEC]:
    a. *Key Block*
    b. *Key Value*
5. Supports the following *Operations Data Structures* [KMIP-SPEC]:
    a. *Capability Information*
    b. *Defaults Information*
    c. *Extension Information*
    d. *Profile Information*
    e. *RNG Parameters*
    f. *Server Information*
    g. *Validation Information*
6. Supports the following *Object Attributes* [KMIP-SPEC]:
    a. *Activation Date*
    b. *Alternative Name*
    c. *Always Sensitive*
    d. *Comment*
    e. *Compromise Date*
    f. *Compromise Occurrence Date*
    g. *Cryptographic Algorithm*
    h. *Cryptographic Length*
    i. *Cryptographic Parameters*
    j. *Cryptographic Usage Mask*
    k. *Deactivation Date*
    l. *Description*
    m. *Digest*
    n. *Extractable*
    o. *Fresh*
    p. *Initial Date*
    q. *Key Value Location*
    r. *Key Value Present*
    s. *Last Change Date*
    t. *Lease Time*
    u. *Link*
    v. *Name*
    w. *Never Extractable*
    x. *Object Group*
    y. *Object Type*
    z. *Original Creation Date*
    aa. *Process Start Date*
    bb. *Protect Stop Date*
    cc. *Protection Storage Mask*
    dd. *Random Number Generator*

ee. *Revocation Reason*
ff. *Sensitive*
gg. *State*
hh. Usage Limits
ii. *Unique Identifier*

7. Supports the following *Client-to-Server Operations* [KMIP-SPEC]
    a. *Activate*
    b. *Add Attribute*
    c. Adjust Attribute
    d. *Check*
    e. *Delete Attribute*
    f. *Destroy*
    g. *Discover Versions*
    h. Export
    i. *Get*
    j. *Get Attribute List*
    k. *Get Attributes*
    l. Import
    m. *Interop*
    n. *Modify Attribute*
    o. Locate
    p. Log
    q. *Query*
    r. Register
    s. *Revoke*
    t. Set Attribute
    u. Set Endpoint Role
8. Supports *Server-to-Client Operations* [KMIP-SPEC]
    a. Discover Versions
    b. Notify
    c. Put
    d. Query
    e. Set Endpoint Role
9. Supports the following *Message Data Structures* [KMIP-SPEC]:
    a. Asynchronous Indicator
    b. Attestation Capable Indicator
    c. Batch Count
    d. Batch Error Continuation Option
    e. Batch Item
    f. Batch Order Option
    g. Client Correlation Value
    h. Maximum Response Size
    i. Message Extension
    j. Operation
    k. Protocol Version
    l. Result Reason
    m. Result Status
    n. Server Correlation Value
    o. Time Stamp
    p. Unique Batch Item ID
10. Supports the *Message Protocols* [KMIP-SPEC]
    a. *Authentication*
    b. *Transport*
    c. TTLV

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

5.1.3 Baseline Mandatory Test Cases KMIP v2.0

5.1.3.1 BL-M-1-20

See test-cases/kmip-v2.0/mandatory/BL-M-1-20.xml.

5.1.3.2 BL-M-2-20

See test-cases/kmip-v2.0/mandatory/BL-M-2-20.xml.

5.1.3.3 BL-M-3-20

See test-cases/kmip-v2.0/mandatory/BL-M-3-20.xml.

5.1.3.4 BL-M-4-20

See test-cases/kmip-v2.0/mandatory/BL-M-4-20.xml.

5.1.3.5 BL-M-5-20

See test-cases/kmip-v2.0/mandatory/BL-M-5-20.xml.

5.1.3.6 BL-M-6-20

See test-cases/kmip-v2.0/mandatory/BL-M-6-20.xml.

5.1.3.7 BL-M-7-20

See test-cases/kmip-v2.0/mandatory/BL-M-7-20.xml.

5.1.3.8 BL-M-8-20

See test-cases/kmip-v2.0/mandatory/BL-M-8-20.xml.

5.1.3.9 BL-M-9-20

See test-cases/kmip-v2.0/mandatory/BL-M-9-20.xml.

5.1.3.10 BL-M-10-20

See test-cases/kmip-v2.0/mandatory/BL-M-10-20.xml.

5.1.3.11 BL-M-11-20

See test-cases/kmip-v2.0/mandatory/BL-M-11-20.xml.

5.1.3.12 BL-M-12-20

See test-cases/kmip-v2.0/mandatory/BL-M-12-20.xml.

5.1.3.13 BL-M-13-20

See test-cases/kmip-v2.0/mandatory/BL-M-13-20.xml.

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:

2. Supports *KMIP Server Implementation Conformance* [KMIP-SPEC]
3. Supports *Objects* [KMIP-SPEC]
4. Supports *Object Data Structures* [KMIP-SPEC]
5. Supports *Object Attributes* [KMIP-SPEC]
6. Supports *Attribute Data Structures* [KMIP-SPEC]
7. Supports *Operations* [KMIP-SPEC]
8. Supports *Operations Data Structures* [KMIP-SPEC]
9. Supports *Messages* [KMIP-SPEC]
10. Supports *Message Data Structures* [KMIP-SPEC]
11. Supports *Message Protocols* [KMIP-SPEC]
12. Supports *Enumerations* [KMIP-SPEC]
13. Supports *Bit Masks* [KMIP-SPEC]
14. 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 v2.0

5.3.3.1 MSGENC-HTTPS-M-1-20

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-v2.0/mandatory/MSGENC-HTTPS-M-1-20.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 XML-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 a CamelCase representation of one of the normalized values as defined in the KMIP specification.

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

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"/>
```

The value SHALL always be “timezoned” – i.e. a timezone specifier SHALL always be included.

5.4.1.6.12 Interval

XML uses [XML-SCHEMA] type xsd:unsignedInt

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

5.4.1.6.13 Date-Time Extended

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

```
<x540001 type="DateTimeExtended" value="2"/>
<X540001 type="DateTimeExtended" value="1152921504606846976"/>
```

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 v2.0

5.4.4.1 MSGENC-XML-M-1-20

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-v2.0/mandatory/MSGENC-XML-M-1-20.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:

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

5.5.1.2 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 a CamelCase representation of one of the normalized values as defined in the KMIP specification.

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

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 of the 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"}
```

The value SHALL always be "timezoned" – i.e. a timezone specifier SHALL always be included.

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"}
```

5.5.1.6.13 Date Time Extended

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":"DateTimeExtended", "value":"0xfffffffffffffffe"}
{"tag": "0x540001", "type":"DateTimeExtended", "value":2}
```

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

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. 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.

5.5.3 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 v2.0

5.5.4.1 MSGENC-JSON-M-1-20

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-v2.0/mandatory/MSGENC-JSON-M-1-20.xml.

The normative corresponding wire encoding in JSON 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":"0x00000004"}
    ]},
    {"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":"0x00000004"}
    ]},
    {"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":"0x00000004"}
    ]},
    {"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":"0x00000004"}
    ]},
    {"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*
3. SHALL support the following *Object Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm*
    b. Object Type
    c. *Process Start Date*
    d. *Protect Stop Date*
4. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC:
    a. *Create*
5. SHALL support the following *Enumerations* [KMIP-SPEC]:
    a. *Cryptographic Algorithm* with values:
        i. 3DES
        ii. AES
    b. *Object Type* with value:
        i. Symmetric Key

c. *Key Format Type* 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 5.6.2
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 v2.0

5.6.3.1 SKLC-M-1-20

See test-cases/kmip-v2.0/mandatory/SKLC-M-1-20.xml.

5.6.3.2 SKLC-M-2-20

See test-cases/kmip-v2.0/mandatory/SKLC-M-2-20.xml.

5.6.3.3 SKLC-M-3-20

See test-cases/kmip-v2.0/mandatory/SKLC-M-3-20.xml.

5.6.4 Symmetric Key Lifecycle Optional Test Cases KMIP v2.0

5.6.4.1 SKLC-O-1-20

See test-cases/kmip-v2.0/optional/SKLC-O-1-20.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*
3. SHALL support the following *Object Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm*
    b. *Cryptographic Length with values:*
        i. *168 (3DES)*
        ii. *128 (AES)*
        iii. *192 (AES*
        iv. *256 (AES)*
    c. *Object Type*
    d. *Process Start Date*
    e. *Protect Stop Date*
4. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC:
    a. *Create*
5. SHALL support the following *Enumerations* [KMIP-SPEC]:
    a. *Cryptographic Algorithm* with values:
        i. *3DES*
        ii. *AES*
    b. *Key Format Type* with value:
        i. *Raw*
        ii. *Transparent Symmetric Key*
    c. *Object Type* with value:
        i. *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 v2.0

5.7.5.1 SKFF-M-1-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-1-20.xml.

5.7.5.2 SKFF-M-2-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-2-20.xml.

5.7.5.3 SKFF-M-3-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-3-20.xml.

5.7.5.4 SKFF-M-4-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-4-20.xml.

5.7.6 Intermediate Symmetric Key Foundry Mandatory Test Cases KMIP v2.0

5.7.6.1 SKFF-M-5-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-5-20.xml.

5.7.6.2 SKFF-M-6-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-6-20.xml.

5.7.6.3 SKFF-M-7-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-7-20.xml.

5.7.6.4 SKFF-M-8-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-8-20.xml.

5.7.7 Advanced Symmetric Key Foundry Mandatory Test Cases KMIP v2.0

5.7.7.1 SKFF-M-9-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-9-20.xml.

5.7.7.2 SKFF-M-10-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-10-20.xml.

5.7.7.3 SKFF-M-11-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-11-20.xml.

5.7.7.4 SKFF-M-12-20

See test-cases/kmip-v2.0/mandatory/SKFF-M-12-20.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. *Public Key*
    b. *Private Key*
3. SHALL support the following *Object Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm*
    b. *Object Type*
    c. *Process Start Date*
    d. *Process Stop Date*
4. SHALL support the following *Enumerations* [KMIP-SPEC]:
    a. *Cryptographic Algorithm* with values:
        i. RSA
    b. *Key Format Type* with value:
        i. PKCS#1
        ii. PKCS#8
        iii. Transparent RSA Public Key
        iv. Transparent RSA Private Key
    c. *Object Type* with value:
        i. Public Key
        ii. Private Key
5. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.8.2
6. 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 v2.0

5.8.3.1 AKLC-M-1-20

See test-cases/kmip-v2.0/mandatory/AKLC-M-1-20.xml.

5.8.3.2 AKLC-M-2-20

See test-cases/kmip-v2.0/mandatory/AKLC-M-2-20.xml

5.8.3.3 AKLC-M-3-20

See test-cases/kmip-v2.0/mandatory/AKLC-M-3-20.xml

5.8.4 Asymmetric Key Lifecycle Optional Test Cases KMIP v2.0

5.8.4.1 AKLC-O-1-20

See test-cases/kmip-v2.0/optional/AKLC-O-1-20.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 Operations* [KMIP-SPEC]:
    a. *Decrypt*
    b. *Encrypt*
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 Operations* [KMIP-SPEC]:
    a. *Decrypt*
    b. *Encrypt*
    c. *Hash*
    d. *MAC*
    e. *MAC Verify*
    f. *RNG Retrieve*
    g. *RNG Seed*
    h. *Sign*
    i. *Signature Verify*
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 Operations* [KMIP-SPEC]:
    a. *RNG Retrieve*
    b. *RNG Seed*
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 Operations* [KMIP-SPEC]:
    a. *Decrypt*
    b. *Encrypt*
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 Operations* [KMIP-SPEC]:
    a. *Decrypt*
    b. *Encrypt*
    c. *Hash*
    d. *MAC*
    e. *MAC Verify*
    f. *RNG Retrieve*
    g. *RNG Seed*
    h. *Sign*
    i. *Signature Verify*
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 Operations* [KMIP-SPEC]:
    a. *RNG Retrieve*
    b. *RNG Seed*
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 v2.0

5.9.7.1 CS-BC-M-1-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-1-20.xml.

5.9.7.2 CS-BC-M-2-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-2-20.xml.

5.9.7.3 CS-BC-M-3-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-3-20.xml.

5.9.7.4 CS-BC-M-4-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-4-20.xml.

5.9.7.5 CS-BC-M-5-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-5-20.xml.

5.9.7.6 CS-BC-M-6-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-6-20.xml.

5.9.7.7 CS-BC-M-7-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-7-20.xml.

5.9.7.8 CS-BC-M-8-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-8-20.xml.

5.9.7.9 CS-BC-M-9-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-9-20.xml.

5.9.7.10 CS-BC-M-10-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-10-20.xml.

5.9.7.11 CS-BC-M-11-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-11-20.xml.

5.9.7.12 CS-BC-M-12-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-12-20.xml.Í

5.9.7.13 CS-BC-M-13-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-20-20.xml.

5.9.7.14 CS-BC-M-14-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-20-20.xml.

5.9.7.15 CS-BC-M-GCM-1-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-GCM-1-20.xml.

5.9.7.16 CS-BC-M-GCM-2-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-GCM-2-20.xml

5.9.7.17 CS-BC-M-GCM-3-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-GCM-3-20.xml.

5.9.7.18 CS-BC-M-CHACHA20-1-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-CHACHA20-1-20.xml.

5.9.7.19 CS-BC-M-CHACHA20-2-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-CHACHA20-2-20.xml.

5.9.7.20 CS-BC-M-CHACHA20-3-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-CHACHA20-3-20.xml.

5.9.7.21 CS-BC-M-CHACHA20POLY1305-1-20

See test-cases/kmip-v2.0/mandatory/CS-BC-M-CHACHA20POLY1305-1-20.xml.

5.9.8 Advanced Cryptographic Mandatory Test Cases KMIP v2.0

5.9.8.1 CS-AC-M-1-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-1-20.xml.

5.9.8.2 CS-AC-M-2-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-2-20.xml.

5.9.8.3 CS-AC-M-3-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-3-20.xml.

5.9.8.4 CS-AC-M-4-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-4-20.xml.

5.9.8.5 CS-AC-M-5-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-5-20.xml.

5.9.8.6 CS-AC-M-6-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-6-20.xml.

5.9.8.7 CS-AC-M-7-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-7-20.xml.

5.9.8.8 CS-AC-M-8-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-8-20.xml.

5.9.8.9 CS-AC-M-OAEP-1-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-1-20.xml.

5.9.8.10 CS-AC-M-OAEP-2-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-2-20.xml.

5.9.8.11 CS-AC-M-OAEP-3-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-3-20.xml

5.9.8.12 CS-AC-M-OAEP-4-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-4-20.xml.

5.9.8.13 CS-AC-M-OAEP-5-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-5-20.xml.

5.9.8.14 CS-AC-M-OAEP-6-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-6-20.xml.

5.9.8.15 CS-AC-M-OAEP-7-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-7-20.xml.

5.9.8.16 CS-AC-M-OAEP-8-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-8-20.xml.

5.9.8.17 CS-AC-M-OAEP-9-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-9-20.xml.

5.9.8.18 CS-AC-M-OAEP-10-20

See test-cases/kmip-v2.0/mandatory/CS-AC-M-OAEP-10-20.xml.

5.9.9 RNG Cryptographic Mandatory Test Cases KMIP v2.0

5.9.9.1 CS-RNG-M-1-20

See test-cases/kmip-v2.0/mandatory/CS-RNG-M-1-20.xml.

5.9.10 RNG Cryptographic Optional Test Cases KMIP v2.0

5.9.10.1 CS-RNG-O-1-20

See test-cases/kmip-v2.0/optional/CS-RNG-O-1-20.xml

5.9.10.2 CS-RNG-O-2-20

See test-cases/kmip-v2.0/optional/CS-RNG-O-2-20.xml

5.9.10.3 CS-RNG-O-3-20

See test-cases/kmip-v2.0/optional/CS-RNG-O-3-20.xml

5.9.10.4 CS-RNG-O-4-20

See test-cases/kmip-v2.0/optional/CS-RNG-O-4-20.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*
3. SHALL support the following *Object Attributes* [KMIP-SPEC]
    a. Object Type
4. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC]:
    a. *Register*
5. SHALL support the following *Enumerations* [KMIP-SPEC]:
    a. *Opaque Data Type*
    b. *Object Type* 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 v2.0

5.10.3.1 OMOS-M-1-20

See test-cases/kmip-v2.0/mandatory/OMOS-M-1-20.xml.

5.10.4 Opaque Managed Object Optional Test Cases KMIP v2.0

5.10.4.1 OMOS-O-1-20

See test-cases/kmip-v2.0/optional/OMOS-O-1-20.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. *Secret Data*
3. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Vendor Attribute*
4. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC]:
    a. *Register*
5. SHALL support the following *Enumerations* [KMIP-SPEC]::
    a. *Name Type* value:
        i. Uninterpreted Text String
    b. *Object Type* values:
        i. Secret Data
    c. *Secret Data Type* value:
        i. Password
6. SHALL support *Vendor Attribute* [KMIP-SPEC] with the following data types and properties:
    a. TextString
7. SHALL support a minimum length of 128 characters for *Vendor Attributes* [KMIP-SPEC] and *Name* [KMIP-SPEC] values where the attribute type is of variable length.
8. SHALL support a minimum of 20 *Vendor Attributes* [KMIP-SPEC] per managed object
9. SHALL support a minimum of 128 characters in *Vendor Attributes* [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 v2.0

5.11.3.1 SASED-M-1-20

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-v2.0/mandatory/SASED-M-1-20.xml.

5.11.3.2 SASED-M-2-20

The secret data for the authentication key is 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-v2.0/mandatory/SASED-M-2-20.xml.

5.11.3.3 SASED-M-3-20

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

See test-cases/kmip-v2.0/mandatory/SASED-M-3-20.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

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 text 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 a text format consisting of hexadecimal numeric character pairs as follows:

1. The unauthenticated KAD is converted to text format by converting each byte value to exactly two Hexadecimal Numeric Characters;
2. The authenticated KAD is converted to text format by converting each byte value to exactly two Hexadecimal Numeric Characters and;
3. The converted authenticated KAD Hexadecimal Numeric Characters are concatenated to the end of the converted unauthenticated KAD Hexadecimal Numeric Characters.

If the implementation uses client-created key identifiers, then the client generates a new identifier in text 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 *Vendor Attributes* [KMIP-SPEC] that duplicates information that is already in standard *Attributes* [KMIP-SPEC]
4. MAY use x-Barcode as a *Vendor Attribute* [KMIP-SPEC] of type *Text String* to store the barcode
5. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Alternative Name*
6. SHALL support the following *Enumerations* [KMIP-SPEC]:
    a. *Alternative Name Type* with 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*
3. SHALL support the following *Attributes* [KMIP-SPEC]:
    a. *Alternative Name*
    b. *Application Specific Information*
    c. *Cryptographic Algorithm*
    d. *Name*
    e. *Vendor Attribute*
4. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC]:
    a. *Create*
5. SHALL support the following *Message Data Structures* [KMIP-SPEC]:
    a. *Batch Coun*t value:
        i. 1 to 32
    b. *Batch Order Option* value:
        i. True
6. SHALL support the following *Enumerations* [KMIP-SPEC]:
    a. Alternative Name Type value:
        i. Uninterpreted Text String
    b. *Cryptographic Algorithm* value:
        i. AES
    c. *Cryptographic Length* value :
        i. *256*
    d. *Key Format Type* value:
        i. Raw
    e. *Name Type* value:
        i. *Uninterpreted Text String*
    f. *Object Type* value:
        i. Symmetric Key
7. SHALL support *Vendor Attributes* [KMIP-SPEC] with the following data types and properties:
    a. Date Time
    b. Integer
    c. Text String
8. SHALL support a minimum length of 255 characters for *Vendor Attributes* [KMIP-SPEC] and *Name* [KMIP-SPEC] values where the attribute type is of variable length
9. SHALL support a minimum of 30 *Vendor Attributes* [KMIP-SPEC] per managed object
10. SHALL support a minimum of 64 characters in *Vendor Attributes* [KMIP-SPEC] names
11. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.12.5
12. 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 v2.0

5.12.6.1 TL-M-1-20

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-v2.0/mandatory/TL-M-1-20.xml.

5.12.6.2 TL-M-2-20

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-v2.0/mandatory/TL-M-2-20.xml.

5.12.6.3 TL-M-3-20

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-v2.0/mandatory/TL-M-3-20.xml.

5.13 AES XTS Profiles

The AES XTS Profile is a KMIP server performing AES XTX key generation related operations based on requests received from a KMIP client.

5.13.1 AES XTS 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.

5.13.2 AES XTS 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*
3. SHALL support the following *Attributes* [KMIP-SPEC]
    a. Object Type
4. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC]:
    a. *Create*
5. SHALL support the following *Enumerations* [KMIP-SPEC]:
    d. *Cryptographic Algorithm* with values:
        i. AES
    e. *Key Format Type* with value:
        i. Raw
        ii. *Transparent Symmetric Key*
    f. *Object Type* with value:
        i. Symmetric Key
6. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.13.2
7. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.13.3 AES XTS Mandatory Test Cases KMIP v2.0

5.13.3.1 AX-M-1-20

Usage of AES XTS directly without a key encrypting key (KEK).

See test-cases/kmip-v2.0/mandatory/AX-M-1-20.xml.

5.13.3.2 AX-M-2-20

Usage of AES XTS directly with a key encrypting key (KEK).

See test-cases/kmip-v2.0/mandatory/AX-M-2-20.xml.

5.14 Quantum Safe Profiles

5.15 Quantum Safe Client

KMIP clients conformant to this profile under [KMIP-SPEC]:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL support `TLS v1.3 [RFC8446]`
3. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section
4. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.16 Quantum Safe Server

KMIP servers conformant to this profile under [KMIP-SPEC]:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL support `TLS v1.3 [RFC8446]`
3. SHALL support the following *Objects* [KMIP-SPEC]
    a. *Certificate*
    b. *Private Key*
    c. *Public Key*
    d. *Symmetric Key*
4. SHALL support the following *Attributes* [KMIP-SPEC]
    a. *Cryptographic Algorithm*
    b. *Cryptographic Length*
    c. *Protection Level*
    d. *Protection Period*
    e. *Quantum Safe*
5. SHALL support the following C*lient-to-Server Operations* [KMIP-SPEC]:
    a. *Certify*
    b. *Create*
    c. *Create Key Pair*
    d. *Decrypt*
    e. *Encrypt*
    f. *Re-Certify*
    g. *Register*
    h. *Re-key*
    i. *Re-key Key Pair*
    j. *Sign*
    k. *Signature Verify*
6. SHALL support the following *Server-to-Client Operations* [KMIP-SPEC]:
    a. *Notify*
    b. Put
7. SHALL support the following *Enumerations* [KMIP-SPEC]:
    a. *Recommended Curve* value:

i. *P-384 (SECP384R1)*
ii. *P-521*

b. *Certificate Type* value:
i. X.*509*

c. *Cryptographic Algorithm* value:
i. *AES*
ii. *ChaCha20*
iii. *ChaChar20Poly1305*
iv. *McEliece-6960119*
v. *McEliece-8192128*
vi. *SPHINCS-256*

d. *Hashing Algorithm* value:
i. *SHA-384*
ii. *SHA-512*
iii. *SHA3-256*
iv. *SHA3-384*
v. *SHA3-512*

e. *Object Type* value:
i. *Certificate*
ii. *Private Key*
iii. *Public Key*
iv. *Symmetric Key*

f. *Key Format Type* value:
i. *Raw*
ii. *X.509*

g. *Digital Signature Algorithm* value:
i. *ECDSA with SHA384 (on P-384)*
ii. *Ed25519* with *Ed25519*

8. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.14.
9. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not conflict with any KMIP requirements.

5.17 Mandatory Quantum Safe Test Cases KMIP v2.0

This section documents the test cases that a client or server conformant to this profile SHALL support.

5.17.1 QS-M-1-12 - Query

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

The specific list of operations noted in the test case are the minimum list of operations – additional operations MAY be supported.

The TLS protocol version and cipher suite SHALL be `TLS v1.3 [RFC8446].`

See test-cases/kmip-v2.0/mandatory/QS-M-1-20.xml.

5.17.2 QS-M-2-20 - Create

Perform a Create operation, stating the period the key must be able to offer protection (Protection Period) and the relative sensitivity of the information (Protection Type).

The TLS protocol version and cipher suite SHALL be `TLS v1.3 [RFC8446].`

See test-cases/kmip-v2.0/mandatory/QS-M-2-20.xml.

5.18 PKCS#11 Profiles

The PKCS profile specifies the use of KMIP to encapsulate PKCS#11 calls.

5.18.1 PKCS#11 Encoding

PKCS#11 function calls are mapped into a KMIP PKCS#11 operation. The parameters are a direct representation of the parameters of the PKCS#11 functions, attributes and other structures defined in [PKCS#11],

The function return values are provided directly in the PKCS#11 Return Code field in the Response Payload. Output Parameters may be omitted in the case of an error status.

For scalar types, CK_BYTE is represented as a single byte, while CK_ULONG and CK_LONG values are transmitted as 8 bytes, network byte order (big-endian). Other PKCS#11 types are built upon those base types as defined in [PKCS#11]. For example, a CK_DATE is a four-byte year followed by a two-byte month and a two-byte day, each byte representing an ASCII character. Strings are either space padded or null terminated as defined in [PKCS#11].

If a parameter or element of a structure represents a pointer to a structure, then the elements of that structure are inserted in line, with its input or output elements listed recursively. If a parameter represents a fixed length array, then the elements appear in order. If the length is variable, then the count of the number of elements is provided immediately before the array, and removed from its original position in the parameter list or structure. CK_ULONGs that represent a count or length are represented as 4 bytes big-endian values.

PKCS#11 functions that handle variable length data structures use a pattern in which the caller first calls with null pointers and the library then fills in the required length. This enables the caller to allocate sufficient memory for the result and call the function a second time with non-null pointers. An additional byte is inserted before such parameters in requests to indicate whether the values are required, or just the lengths.

Templates are represented by a 4 byte big-endian count of attributes followed by the attributes themselves. They may be encoded with or without the values for C_GetAttributeValue which reflects whether it has been called with null pointers or has maximum lengths already determined.

For each attribute a one byte value indicator flag that indicates whether a value was defined via the pValue field of the attribute. This is followed by a second one byte count indicator flag that indicates if a multiple of the count of values was defined via the ulValueLen field of the attribute. If the count indicator is set to false (0), then the value indicator SHALL be set to false (0) also. In the case of a C_GetAttributeValue input request, the value indicator SHALL be set to false (0) unless the value is an array attribute other than CKA_ALLOWED_MECHANISMS. .

Fixed length attributes (attributes of type CK_ULONG, CK_BYTE, CK_BOOL, CK_CHAR and fixed length arrays of those types) are provided in line, with an implicit count of the number of elements which is not present in the encoding. Byte strings are also provided in line, but preceded by a 4 byte big-endian count of the number of bytes. CKA_ALLOWED_MECHANISMS attribute values are preceded by a count of the number of elements followed by the values themselves; all other array attributes are stored recursively in the same manner as the encapsulating template.

Mechanisms are encoded by an 8 byte big-endian mechanism number followed by a one byte flag that indicates whether the parameter field follows. If the flag is set (to 1), it is followed by a 4 byte big-endian field that stores the length in bytes of any mechanism parameter followed by the parameter itself. If the parameter is a byte string such as an Initialization Vector it is preceded by a second 4 byte big-endian length. The fields in structured parameters such as CK_GCM_PARAMS are simply stored sequentially, with any contained byte strings also preceded by a 4 byte big-endian length.

Values and functions that are only meaningful to the API itself are not encapsulated, nor are void pointers that do not have any well-defined meaning. In particular:-

- C_Initialize does not pass through the pInitArgs structure. (It contains pointers to multi-threading functions.)
- C_Initialize passes a single BYTE parameter that encodes the version of this encoding. The version SHALL be set to 1 for implementations conformant to this profile.
- C_Finalize does not pass through its parameter.
- C_GetFunctionList is not used. A PKCS#11 stub is expected to provide the function list without reference to the KMIP server.
- C_GetInterface is not used. A PKCS#11 stub is expected to provide the interface without reference to the KMIP server.
- C_WaitForSlotEvent is not used.
- Callbacks are not supported. Accordingly, the Notify parameter on C_OpenSession is not passed to the server.

5.18.2 PKCS#11 Examples

5.18.2.1 PKCS#11 Initialization

```
CK_RV rv;
CK_FUNCTION_LIST_PTR pFunctionList;
CK_C_Initialize pC_Initialize;

rv = C_GetFunctionList(&pFunctionList); /* C_GetFunctionLists in V3.0 */
pC_Initialize = pFunctionList -> C_Initialize;

/* Call the C_Initialize function in the library */
CK_C_INITIALIZE_ARGS InitArgs;

InitArgs.CreateMutex = &MyCreateMutex;
InitArgs.DestroyMutex = &MyDestroyMutex;
InitArgs.LockMutex = &MyLockMutex;
InitArgs.UnlockMutex = &MyUnlockMutex;
InitArgs.flags = CKF_OS_LOCKING_OK;
InitArgs.pReserved = NULL_PTR;

rv = (*pC_Initialize)((CK_VOID_PTR)&InitArgs);

CK_INFO info;
rv = (*(pFunctionList -> C_GetInfo))(&info);
if(info.version.major == 2) {...}
```

```
CK_SLOT_ID pSlotList[64];
CK_ULONG ulSlotCount = 64;
rv = (*(pFunctionList ->C_GetSlotList))(CK_TRUE, pSlotList, ulSlotCount);
```

C_GetFunctionList

Not passed through

Input C_Initialize

```
<PKCS_11Function type="Enumeration" value="C_Initialize"/>
<PKCS_11InputParameters type="Byte String" value= ...
01                      Version of encoding
```

Output C_Initialize

```
<PKCS_11Function type="Enumeration" value="C_Initialize"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
  <!-- No Output parameters -->
```

Input C_GetIInfo

```
<PKCS_11Function type="Enumeration" value="C_GetInfo"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
   <!-- No Input Parameters -->
```

Output C_GetInfo

```
<PKCS_11Function type="Enumeration" value="C_GetInfo"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11OutputParameters type="Byte String" value=
  <!-- Fields defined by CK_INFO structure. -->
0228                    cryptokiVersion
4142...                 manufacturerID 32 bytes, blank filled
0000 0000 0000 0000     flags
4142...                 libraryDescription 32 bytes, blank filled
0101                    libraryVersion
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
```

Input C_GetSlotList

```
<PKCS_11Function type="Enumeration" value="C_GetSlotList"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11InputParameters type="Byte String" value=
01                      CK_TRUE, tokenPresent indicator
01                      Slot info required
0000 0040               64 slots in the request array
```

Output C_GetSlotList


```
<PKCS_11Function type="Enumeration" value="C_GetSlotList"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11OutputParameters type="Byte String" value=
  <!-- Fields defined by parameter list. -->
01                    Slot values are present
0000 0002             ulSlotCount returned
0000 0000 1234 5678   First CK_SLOT_ID
0000 0000 ABCD 0987   Second CK_SLOT_ID
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
```

5.18.2.2 PKCS#11 C_Encrypt

```
#define PLAINTEXT_BUF_SZ 195
#define CIPHERTEXT_BUF_SZ 256

CK_ULONG firstPartLen, secondPartLen;
CK_SESSION_HANDLE hSession = 0x12345678;  /* For example only */
CK_OBJECT_HANDLE hKey = 0x87654321;
CK_BYTE iv[8] = {1 ,2 ,3, 4, 5, 6, 7, 8};
CK_MECHANISM mechanism = {
  CKM_DES_CBC_PAD, iv, sizeof(iv)
};
CK_BYTE data[PLAINTEXT_BUF_SZ] = {01, 02, 03, …};
CK_BYTE encryptedData[CIPHERTEXT_BUF_SZ];
CK_ULONG ulEncryptedData1Len; /* Output only(!) */
CK_ULONG ulEncryptedData2Len;
CK_ULONG ulEncryptedData3Len;

.
.
firstPartLen = 90;
secondPartLen = PLAINTEXT_BUF_SZ-firstPartLen;
C_EncryptInit(hSession, &mechanism, hKey);


/* Encrypt first Part */
ulEncryptedData1Len = sizeof(encryptedData);
C_EncryptUpdate(
    hSession,
    &data[0], firstPartLen,
    &encryptedData[0], &ulEncryptedData1Len);

/* Encrypt second Part */
ulEncryptedData2Len = sizeof(encryptedData)-ulEncryptedData1Len;
C_EncryptUpdate(
    hSession,
    &data[firstPartLen], secondPartLen,
    &encryptedData[ulEncryptedData1Len], &ulEncryptedData2Len);

/* Get last little encrypted bit */
ulEncryptedData3Len =
    sizeof(encryptedData)-ulEncryptedData1Len-ulEncryptedData2Len;
C_EncryptFinal(
    hSession,
    &encryptedData[ulEncryptedData1Len+ulEncryptedData2Len],
    &ulEncryptedData3Len);
```

Input C_EncryptInit

```
<PKCS_11Function type="Enumeration" value="C_EncryptInit"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11InputParameters type="Byte String" value= ...
0000 0000 1234 5678     hSession
0000 0000 0000 0125     CKM_DES_CBC_PAD
01                      Parameter is present
0000 000C               Entire parameter length encoding
0000 0008               ulParameterLen (for simple CK_BYTE array)
0102 0304 0506 0708     pParameter, the IV
0000 0000 8765 4321     hKey
```

Output C_EncryptInit

```
<PKCS_11Function type="Enumeration" value="C_EncryptInit"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
```

Input C_EncryptUpdate 1

```
<PKCS_11Function type="Enumeration" value="C_EncryptUpdate"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11InputParameters type="Byte String" value= ...

0000 0000 1234 5678     hSession
0000 005A               90, firstPartLen
0102 0304 …    595A     90 bytes of plain text
01                      encryptedPart wanted
0000 0100               256 available in encryptedPart buffer
```

Output C_EncryptUpdate 1

```
<PKCS_11Function type="Enumeration" value="C_EncryptUpdate"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11OutputParameters type="Byte String" value= ...
0000 0058               88, SIZE OF the first 11 blocks
A698 C3D8 ...           88 bytes of cipher text
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
```

Input C_EncryptUpdate 2

```
<PKCS_11Function type="Enumeration" value="C_EncryptUpdate"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11InputParameters type="Byte String" value= ...

0000 0000 1234 5678     hSession
0000 006E               105, secondPartLen
5B5C 5D5E ...  C2C3     105 bytes of plain text
01                      encryptedPart wanted
0000 00A8               256 - 88 available in encryptedPart buffer
```

Output C_EncryptUpdate 2

```
<PKCS_11Function type="Enumeration" value="C_EncryptUpdate"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11OutputParameters type="Byte String" value= ...
0000 0058               104, size the second 13 blocks
4A69 5C3D …             104 bytes of cipher text; 1 byte outstanding.
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
```

Input C_EncryptFinal

```
<PKCS_11Function type="Enumeration" value="C_EncryptFinal"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11InputParameters type="Byte String" value= ...
0000 0000 1234 5678     hSession
01                      encryptedPart wanted
0000 0040               256 - 88 - 104 available in encryptedPart buffer
```

Output C_EncryptFinal

```
<PKCS_11Function type="Enumeration" value="C_EncryptFinal"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11OutputParameters type="Byte String" value= ...
0000 0000 1234 5678     hSession
0000 0008               8, size the final block
65BA C53D …             8 bytes of cipher text
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
```

5.18.2.3 PKCS#11 C_GetAttributeValue

```
CK_SESSION_HANDLE hSession = 0x12345678;  /* For example only */
CK_OBJECT_HANDLE hObject = 0x87654321;
CK_BYTE sensitive;
CK_BYTE_PTR checkValue[16];
CKA_MECHANISM_TYPE mechanisms[64];
CK_ATTRIBUTE template[] = {
  {CKA_SENSITIVE, sensitive, sizeof(sensitive)},
  {CKA_CHECK_VALUE, checkValue, sizeof(checkValue)},
  {CKA_ALLOWED_MECHANISMS, mechanisms, sizeof(mechanisms)},
  {CKA_LABEL, NULL_PTR, 42 }
};
CK_RV rv;
rv = C_GetAttributeValue(hSession, hObject, &template, 4);
```

Input

```
<PKCS_11Function type="Enumeration" value="C_GetAttributeValue"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11InputParameters type="Byte String" value= ...
```

```
0000 0000 1234 5678     hSession
0000 0000 8765 4321     hObject
0000 0004               ulCount (Number of templates, moved up)
0000 0000 0000 0103     CKA_SENSITIVE
00                      Value not defined
01                      Length defined
0000 0000 0000 0090     CKA_CHECK_VALUE
00                      Value not defined
01                      Length defined
0000 0000 4000 0600     CKA_ALLOWED_MECHANISMS
00                      Value not defined
01                      Length defined
0000 0040               64 mechanisms available
00                      Value not defined
00                      Length not defined (is output only)
```

Output

```
<PKCS_11Function type="Enumeration" value="C_GetAttributeValue"/>
<CorrelationValue type="ByteString" value="ABCD1234"/>
<PKCS_11OutputParameters type="Byte String" value= ...
0000 0004               ulCount (Number of templates, moved up)
0000 0000 0000 0103     CKA_SENSITIVE
01                      Value defined
01                      Length defined
01                      CK_TRUE
0000 0000 0000 0090     CKA_CHECK_VALUE
01                      Value defined
01                      Length defined
0000 0010               ulValueLen 16 -- number of byte values
1234 ..... ABCD         Check value
0000 0000 4000 0600     CKA_ALLOWED_MECHANISMS
01                      Value defined
01                      Length defined
0000 0002               2 mechanisms
0000 0000 0000 0121     CKM_DES_ECB
0000 0000 0000 0125     CKM_DES_CBC_PAD
0000 0000 0000 0003     CKA_LABEL
00                      Value not defined
01                      Length defined
0000 00003              3 bytes (for an example label of "foo")
<PKCS_11ReturnCode type="Enumeration" value="OK"/>
```

5.18.3 PKCS#11 Client

KMIP clients conformant to this profile:

1. SHALL conform to the Baseline Client (section 5.1.1)
2. SHALL conform with PKCS#11 Encoding (5.18.1)
3. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC]:
    a. *PKCS#11*
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.

5.18.4 PKCS#11 Server

KMIP servers conformant to this profile:

1. SHALL conform to the Baseline Server (section 5.1.2)
2. SHALL conform with PKCS#11 Encoding (5.18.1)
3. SHALL support the following *Client-to-Server Operations* [KMIP-SPEC]:
    a. *PKCS#11*
4. MAY support any clause within [KMIP-SPEC] provided it does not conflict with any other clause within this section 5.5.3
5. MAY support extensions outside the scope of this standard (e.g., vendor extensions, conformance clauses) that do not contradict any KMIP requirements.

5.18.5 PKCS#11 Mandatory Test Cases KMIP v2.0

5.18.5.1 PKCS11-M-1-20

See test-cases/kmip-v2.0/mandatory/PKCS11-M-1-20.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 v2.0 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) and;
4. SHALL support one or more of the Baseline Mandatory Test Cases KMIP v2.0 (5.1.3, 5.6.3).

6.2 Baseline Server Basic KMIP v2.0 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) and;
4. SHALL support all of the Baseline Mandatory Test Cases KMIP v2.0 (5.1.3, 5.6.3).

6.3 Complete Server Basic KMIP v2.0 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) and;
4. SHALL support all of the server conformance clauses contained within Conformance (6)

6.4 HTTPS Client KMIP v2.0 Profile Conformance

KMIP client implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the HTTPS Authentication Suite conditions (3.2) and;
3. SHALL support the HTTPS Client conditions (5.3.1) and;
4. SHALL support all of the HTTPS Mandatory Test Cases KMIP v2.0 (5.3.3); and
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.5 HTTPS Server KMIP v2.0 Profile Conformance

KMIP server implementations conformant to this profile:

1. SHALL support [KMIP-SPEC]
2. SHALL support the HTTPS Authentication Suite conditions (3.2) and;
3. SHALL support the HTTPS Server conditions (5.3.2) and;
4. SHALL support all of the HTTPS Mandatory Test Cases KMIP v2.0 (5.3.3) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.6 XML Client KMIP v2.0 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 v2.0 (5.4.4) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.7 XML Server KMIP v2.0 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 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 v2.0 (5.4.4) and;
6. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.8 JSON Client KMIP v2.0 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 v2.0 (5.5.4) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.9 JSON Server KMIP v2.0 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 v2.0 (5.5.4) and;
6. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.10 Symmetric Key Lifecycle Client KMIP v2.0 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 v2.0 (5.6.3) and;

5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.11 Symmetric Key Lifecycle Server KMIP v2.0 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 v2.0 (5.6.3) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.12 Basic Symmetric Key Foundry Client KMIP v2.0 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 v2.0 (5.7.5) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.13 Intermediate Symmetric Key Foundry Client KMIP v2.0 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 v2.0 (5.7.6) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.14 Advanced Symmetric Key Foundry Client KMIP v2.0 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 v2.0 (5.7.7) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.15 Symmetric Key Foundry Server KMIP v2.0 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 Foundry Server conditions (5.7.4) and;
4. SHALL support all of the Basic Symmetric Key Foundry Mandatory Test Cases KMIP v2.0 (5.7.5) and;
5. SHALL support all of the Intermediate Symmetric Key Foundry Mandatory Test Cases KMIP v2.0 (5.7.6) and;
6. SHALL support all of the Advanced Symmetric Key Foundry Mandatory Test Cases KMIP v2.0 (5.7.7) and;
7. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.16 Asymmetric Key Lifecycle Client KMIP v2.0 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 (5.8.1) and;
4. SHALL support one or more of the Asymmetric Key Lifecycle Mandatory Test Cases KMIP v2.0 (5.8.3) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.17 Asymmetric Key Lifecycle Server KMIP v2.0 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 Server conditions (5.8.2) and;
4. SHALL support all of the Asymmetric Key Lifecycle Mandatory Test Cases KMIP v2.0 (5.8.3) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.18 Basic Cryptographic Client KMIP v2.0 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;
4. SHALL support one or more of the Basic Cryptographic Mandatory Test Cases KMIP v2.0 (5.9.7) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.19 Advanced Cryptographic Client KMIP v2.0 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 v2.0 (5.9.8) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.20 RNG Cryptographic Client KMIP v2.0 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 v2.0 (5.9.9) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.21 Basic Cryptographic Server KMIP v2.0 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;
4. SHALL support all of the Basic Cryptographic Mandatory Test Cases KMIP v2.0 (5.9.7) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.22 Advanced Cryptographic Server KMIP v2.0 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 v2.0 (5.9.8) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.23 RNG Cryptographic Server KMIP v2.0 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 v2.0 (5.9.9) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.24 Opaque Managed Object Client KMIP v2.0 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 v2.0 (5.10.3) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.25 Opaque Managed Object Server KMIP v2.0 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 v2.0 (5.10.3) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.26 Storage Array with Self-Encrypting Drives Client KMIP v2.0 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 v2.0 (5.11.3) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.27 Storage Array with Self-Encrypting Drives Server KMIP v2.0 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 v2.0 (5.11.3) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.28 Tape Library Client KMIP v2.0 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 v2.0 (5.12.6) and;
7. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.29 Tape Library Server KMIP v2.0 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 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 v2.0 (5.12.6) and;
7. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.30 AES XTS Client KMIP v2.0 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 AES XTS Client conditions (5.13.1) and;
4. SHALL support one or more of the AES XTS Mandatory Test Cases KMIP v2.0 (5.13.3) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.31 AES XTS Server KMIP v2.0 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 AES XTS Server conditions (5.13.2) and;
4. SHALL support all of the AES XTS Mandatory Test Cases KMIP v2.0 (5.13.3) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.32 Quantum Safe Client KMIP V2.0 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 Quantum Safe Client conditions (5.15) and;
4. SHALL support one or more of the Mandatory Quantum Safe Test Cases KMIP v2.0 (5.17); and
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.33 Quantum Safe Server KMIP V2.0 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 Quantum Safe Server conditions (5.16) and;
4. SHALL support all of the Mandatory Quantum Safe Test Cases KMIP v2.0 (5.17) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

6.34 PKCS#11 Client KMIP V2.0 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 PKCS#11 Client conditions (5.18.3) and;
4. SHALL support one or more of the PKCS#11 Mandatory Test Cases KMIP v2.0 (5.18.5) and;
5. SHALL support Baseline Client Basic KMIP v2.0 Profile Conformance (6.1)

6.35 PKCS#11 Server KMIP V2.0 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 PKCS#11 Server conditions (5.18.4) and;
4. SHALL support one or more of the PKCS#11 Mandatory Test Cases KMIP v2.0 (5.18.5) and;
5. SHALL support Baseline Server Basic KMIP v2.0 Profile Conformance (6.2)

Appendix A. Acknowledgments

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

Participants:

Rinkesh Bansal - IBM
Jeff Bartell - Individual
Gabriel Becker - KRYPTUS
Andre Bereza - KRYPTUS
Anthony Berglas - Cryptsoft Pty Ltd.
Mathias Bjorkqvist - IBM
Joseph Brand - Semper Fortis Solutions
Alan Brown - Thales e-Security
Andrew Byrne - Dell
Tim Chevalier - NetApp
Kenli Chong - QuintessenceLabs Pty Ltd.
Justin Corlett - Cryptsoft Pty Ltd.
Tony Cox - Cryptsoft Pty Ltd.
James Crossland - Northrop Grumman
Stephen Edwards - Semper Fortis Solutions
Stan Feather - Hewlett Packard Enterprise (HPE)
Indra Fitzgerald - Utimaco IS GmbH
Judith Furlong - Dell
Gary Gardner - Fornetix
Susan Gleeson - Oracle
Steve He - Thales e-Security
Christopher Hillier - Hewlett Packard Enterprise (HPE)
Tim Hudson - Cryptsoft Pty Ltd.
Nitin Jain - SafeNet, Inc.
Gershon Janssen - Individual
Mark Joseph - P6R, Inc
Paul Lechner - KeyNexus Inc
John Leiseboer - QuintessenceLabs Pty Ltd.
Jarrett Lu - Oracle
Jeff MacMillan - KeyNexus Inc
John Major - QuintessenceLabs Pty Ltd.
Cecilia Majorel - Quintessence Labs
Gabriel Mandaji - KRYPTUS
Jon Mentzell - Fornetix
Prashant Mestri - IBM
Kevin Mooney - Fornetix
Ladan Nekuii - Thales e-Security
Jason Novecosky - KeyNexus Inc
Matt O'reilly - Fornetix
Sanjay Panchal - IBM
Mahesh Paradkar - IBM
Steve Pate - Thales e-Security
Greg Pepus - Semper Fortis Solutions
Bruce Rich - Cryptsoft Pty Ltd.
Thad Roemer - Dyadic Security Ltd.
Thad Roemer - Unbound Tech
Greg Scott - Cryptsoft Pty Ltd.
Martin Shannon - QuintessenceLabs Pty Ltd.
Gerald Stueve - Fornetix

Jim Susoy - P6R, Inc
Jason Thatcher - Cryptsoft Pty Ltd.
Peter Tsai - Thales e-Security
Charles White - Fornetix
Steven Wierenga - Utimaco IS GmbH
Kyle Wuolle - KeyNexus Inc

Appendix B. Revision History

Revision	Date	Editor	Changes Made
wd06	11-Apr-2019	Tim Hudson	Added participant list
wd05	28-Mar-2019	Tim Hudson	Post-interop corrections and clarifications.
wd03	20-Dec-2018	Tim Hudson	Added PKCS#11 Profile. Added additional test cases to various profiles. Included updated baseline operations list.
wd02	18-Oct-2018	Tim Hudson	Update PQC to Quantum Safe and include initial Quantum Safe algorithms; Order lists alphabetically to match specification approach. Update various items for consistency.
wd01	11-Oct-2018	Tim Hudson	Initial draft