KMIP 64-bit Binary Alignment Proposal

To: OASIS KMIP Technical Committee

From: Matt Ball, Sun Microsystems, Inc.

Date: May 6, 2009

Version: 2

Purpose: To propose a change to the binary encoding such that each part is aligned to an 8-byte boundary

Revision History

- Version 1, 2009-05-01: Initial version
- Version 2, 2009-05-06: Fixed typos (it appears that there is a style in the current draft that turns off "Check spelling and grammar" for certain styles, which is why these typos were missed the first time). Changed length of *boolean* from 4 to 8 bytes.

Introduction

The binary encoding as defined in the 1.0 version of the KMIP draft does not maintain alignment to 8-byte boundaries within the message structure. This causes problems on hard-aligned processors, such as the ARM, that are not able to easily access memory on addresses that are not aligned to 4 bytes. Additionally, it reduces performance on modern 64-bit processors. For hard-aligned processors, when unaligned memory contents are requested, either the compiler has to add extra instructions to perform two aligned memory accesses and reassemble the data, or the processor has to take a 'trap' (i.e., an interrupt generated on unaligned memory accesses) to correctly assemble the memory contents. Either of these options results in reduced performance. On soft-aligned processors, the hardware has to make two memory accesses instead of one when the contents are not properly aligned.

This proposal suggests ways to improve the performance on hard-aligned processors by aligning all data structures to 8-byte boundaries.

Summary of Proposed Changes

This proposal includes the following changes to the KMIP 0.98 draft submission to the OASIS KMIP TC:

- Change the alignment of the KMIP binary encoding such that each part is aligned to an 8-byte boundary. This is done by:
  - Change the Tag field to occupy 3 bytes. In this way, the combined size of the Tag, Type, and Length fields is 8 bytes.
  - Require that all Item Value fields be padded with zero to seven bytes of the value 00 such that the length of the Value field is a multiple of 8 bytes. The Item Length still contains the correct, unpadded length of the Item.
- Change the length of the Item Value for Binary types to be 4 bytes (instead of 1 byte), with hex values 00000000 for false and 00000001 for true.
- Change the format of the Big Integer Item Type to require that the Item Value be padded with sign-extended bytes on the left (i.e., most significant bytes) such that the total length is a multiple of 8 bytes.

Proposed Changes

The following text shows proposed changes to the KMIP 1.0 draft as published on April 30, 2009, with underline to indicate additions and to indicate deletions.

9 Message Encoding

To support different transport protocols and different client capabilities, a number of message-encoding mechanisms are supported.

9.1 TTLV Encoding

In order to minimize the resource impact on potentially low-function clients, one encoding mechanism to be used for protocol messages is a simplified TTLV (Tag, Type, Length, Value) scheme.

The scheme is designed to minimize the CPU cycle and memory requirements of clients that must encode or decode protocol messages, and to provide optimal alignment for both 32-bit and 64-bit processors. Minimizing bandwidth over the transport mechanism is considered to be of lesser importance.

9.1.1 TTLV Encoding Fields

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

9.1.1.1 Item Tag

An Item Tag is a 3-byte binary unsigned integer, transmitted big endian, which contains a number that designates the specific Protocol Field or Object that the TTLV object represents. To ease debugging, and to ensure that malformed messages are detected more easily, all tags must contain either the value 42 in hex or the value 54 in hex as the high order (first) byte. Tags defined by this specification contain hex 42 in the first byte. Extensions, which are permitted, but not defined in this specification, contain the value 54 hex in the first byte. A list of defined Item Tags is in Section 9.1.3.1 .

9.1.1.2 Item Type

An Item Type is a byte containing a coded value that indicates the data type of the data object. The allowed values are:

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

9.1.1.3 Item Length

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

Allowed values are:

Data Type	Length
Integer	4
Long Integer	8
Big Integer	Varies, multiple of 8
Enumeration	4
Boolean	8
Text String	Varies
Octet String	Varies
Date-Time	8
Interval	4
Structure	Varies, multiple of 8

If the Item Type is a Text String or Octet String, then the Item Length must be the number of unpadded bytes in the string. Strings must not be null-terminated, but must be padded such that the transmitted length of the Item Value is a multiple of 8 bytes (see 9.1.1.4). If the Item Type is a structure, then the Item Length is the total length of all of the sub-items contained in the structure, including any padding. If the Item Type is a Big Integer, then

the Item Length must be the number of bytes in the string after padding with leading bytes to make the length a multiple of 8 bytes.

9.1.1.4 Item Value

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

- Integers are encoded as 4-byte long (32 bit) binary signed numbers in 2's complement notation, transmitted big-endian.
- Long Integers are encoded as 8-byte long (64 bit) binary signed numbers in 2's complement notation, transmitted big-endian.
- Big Integers are encoded as a sequence of 8-bit bytes, in 2's complement notation, transmitted big-endian. If the length is not a multiple of 8 bytes, then the Big Integer shall be padded with enough leading sign-extended bytes to make the length a multiple of 8 bytes.
- Enumerations are encoded as 4-byte long (32 bit) binary unsigned numbers transmitted big-endian. Extensions, which are permitted, but not defined in this specification, contain the value 8 hex in the first nibble of the first byte.
- Booleans are encoded as an 8-byte value that must either contain the hexadecimal value 00000000 00000000, indicating the boolean value *False,* or the hexadecimal value 00000000 00000001, transmitted big-endian, indicating the boolean value *True.*
- Text Strings are sequences of bytes encoding character values according to the UTF-8 encoding standard. There must be no null-termination at the end of such strings.
- Octet Strings are sequences of bytes containing individual unspecified 8 bit binary values.
- Date-Time values are encoded as 8-byte long (64 bit) binary signed numbers, transmitted big-endian. They are POSIX Time values (described in IEEE Standard 1003.1) extended to a 64 bit value to eliminate the “Year 2038 problem”. The value is expressed as the number of seconds from a time epoch, which is 00:00:00 GMT January 1st, 1970. This value has a resolution of 1 second. All Date-Time values are expressed as UTC values.
- Intervals are encoded as 4-byte long (32 bit) binary unsigned numbers, transmitted big-endian. They have a resolution of 1 second.
- Structure Values are encoded as the concatenated encodings of the elements of the structure. All structures defined in this specification must have all of their fields encoded in the order in which they appear in their respective structure descriptions.

9.1.2 If the Item Length contains a value that is not a multiple of 8, then the contents of the Item Value shall be padded with the minimal quantity of trailing ‘00’ bytes such that the transmitted length is a multiple of 8 bytes.Examples

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

- An Integer containing the decimal value 8:

  ```
  42 00 20 | 02 | 00 00 00 04 | 00 00 00 08 00 00 00 00
  ```

- A Long Integer containing the decimal value 123456789000000000:

  ```
  42 00 20 | 03 | 00 00 00 08 | 01 B6 9B 4B A5 74 92 00
  ```

- A Big Integer containing the decimal value 1234567890000000000000000000:

```
42 00 20 | 04 | 00 00 00 10 | 00 00 00 00 03 FD 35 EB 6B C2 DF 46
18 08 00 00
```

- An Enumeration with value 255:

```
42 00 20 | 05 | 00 00 00 04 | 00 00 00 FF 00 00 00 00
```

- A Boolean with the value *True*:

```
42 00 20 | 06 | 00 00 00 04 | 00 00 00 00 00 00 00 01
```

- A Text String:

```
42 00 20 | 07 | 00 00 00 0B | 48 65 6C 6C 6F 20 57 6F 72 6C 64 00
00 00 00 00
```

- An Octet String:

```
42 00 20 | 08 | 00 00 00 03 | 01 02 03 00 00 00 00 00
```

- A Date-Time, containing the value for Friday, March 14, 2008, 11:56:40 GMT:

```
42 00 20 | 09 | 00 00 00 08 | 00 00 00 00 47 DA 67 F8
```

- An Interval, containing the value for 10 days:

```
42 00 20 | 0A | 00 00 00 04 | 00 0D 2F 00 00 00 00 00
```

- A Structure containing an Enumeration, value 254, followed by an Integer, value 255, having tags 42000004 and 42000005 respectively:

```
42 00 20 | 01 | 00 00 00 20 | 42 00 04 | 05 | 00 00 00 04 | 00 00
00 FE 00 00 00 00 | 42 00 05 | 02 | 00 00 00 04 | 00 00 00 FF 00
00 00 00
```

9.1.3 Defined Values

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

9.1.3.1 Tags

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

Tag	
Object	**Tag Value**
(Unused)	`000000 - 420000`
Activation Date	`420001`
Application Identifier	`420002`
Application Name Space	`420003`
Application Specific Identification	`420004`
Archive Date	`420005`
Asynchronous Correlation Value	`420006`
Asynchronous Indicator	`420007`
Attribute	`420008`

Tag	
Object	**Tag Value**
Attribute Index	420009
Attribute Name	42000A
Attribute Value	42000B
Authentication	42000C
Batch Count	42000D
Batch Error Continuation Option	42000E
Batch Item	42000F
Batch Order Option	420010
Block Cipher Mode	420011
Cancellation Result	420012
Certificate	420013
Certificate Issuer	420014
Certificate Request	420015
Certificate Request Type	420016
Certificate Subject	420017
Certificate Subject Alternative Name	420018
Certificate Subject Distinguished Name	420019
Certificate Type	42001A
Certificate Value	42001B
Common Template-Attribute	42001C
Compromise Date	42001D
Compromise Occurrence Date	42001E
Contact Information	42001F
Credential	420020
Credential Type	420021
Credential Value	420022
Criticality Indicator	420023
CRT Coefficient	420024
Cryptographic Algorithm	420025
Cryptographic Length	420026
Cryptographic Parameters	420027
Cryptographic Usage Mask	420028
Custom Attribute	420029
D	42002A
Deactivation Date	42002B
Derivation Data	42002C

Tag	
Object	**Tag Value**
Derivation Method	42002D
Derivation Parameters	42002E
Destroy Date	42002F
Digest	420030
Digest Value	420031
Encryption Key Information	420032
G	420033
Hashing Algorithm	420034
Initial Date	420035
Initialization Vector	420036
Issuer	420037
Iteration Count	420038
IV/Counter/Nonce	420039
J	42003A
Key	42003B
Key Block	42003C
Key Material	42003D
Key Part Identifier	42003E
Key Value	42003F
Key Value Type	420040
Key Wrapping Data	420041
Key Wrapping Specification	420042
Last Changed Date	420043
Lease Time	420044
Link	420045
Link Type	420046
Linked Object Identifier	420047
MAC/Signature	420048
MAC/Signature Key Information	420049
Maximum Items	42004A
Maximum Response Size	42004B
Message Extension	42004C
Modulus	42004D
Name	42004E
Name Type	42004F
Name Value	420050
Object Group	420051

Tag	
Object	**Tag Value**
Object Type	420052
Offset	420053
Opaque Data Type	420054
Opaque Data Value	420055
Opaque Object	420056
Operation	420057
Operation Policy Name	420058
P	420059
Padding Method	42005A
Policy Template	42005B
Prime Exponent P	42005C
Prime Exponent Q	42005D
Prime Field Size	42005E
Private Exponent	42005F
Private Key	420060
Private Key Template-Attribute	420061
Private Key Unique Identifier	420062
Process Start Date	420063
Protect Stop Date	420064
Protocol Version	420065
Protocol Version Major	420066
Protocol Version Minor	420067
Public Exponent	420068
Public Key	420069
Public Key Template-Attribute	42006A
Public Key Unique Identifier	42006B
Put Function	42006C
Q	42006D
Q String	42006E
Query Function	42006F
Recommended Curve	420070
Replaced Unique Identifier	420071
Request Header	420072
Request Message	420073
Request Payload	420074
Response Header	420075
Response Message	420076
Response Payload	420077

Tag	
Object	**Tag Value**
Result Message	420078
Result Reason	420079
Result Status	42007A
Revocation Message	42007B
Revocation Reason	42007C
Revocation Reason Code	42007D
Role Type	42007E
Salt	42007F
Secret Data	420080
Secret Data Type	420081
Serial Number	420082
Server Information	420083
Split Key	420084
Split Key Method	420085
Split Key Parts	420086
Split Key Threshold	420087
State	420088
Storage Status Mask	420089
Symmetric Key	42008A
Template	42008B
Template Name	42008C
Template-Attribute	42008D
Time Stamp	42008E
Unique Identifier	42008F
Unique Message ID	420090
Usage Limits	420091
Usage Limits Byte Count	420092
Usage Limits Object Count	420093
Usage Limits Total Bytes	420094
Usage Limits Total Objects	420095
Validity Date	420096
Validity Indicator	420097
Vendor Extension	420098
Vendor Identification	420099
Wrapping Method	42009A
X	42009B
Y	42009C
(Reserved)	42009D – 42FFFF

Tag	
Object	**Tag Value**
(Unused)	430000 – 53FFFF
Extensions	540000 – 54FFFF
(Unused)	550000 - FFFFFF