--------------------------------------------------------------------------------------------------------------------------------------

B1 Coding of the Tag Field of BER-TLV Data Objects

Table 35 describes the first byte of the tag field of a BER-TLV data object:

****

Table 35: Tag Field Structure (First Byte) BER-TLV

According to ISO/IEC 8825, Table 36 defines the coding rules of the subsequent bytes of a BER-TLV tag when tag numbers  31 are used (that is, bits b5 - b1 of the first byte equal '11111').

****

Table 36: Tag Field Structure (Subsequent Bytes) BER-TLV

Before, between, or after TLV-coded data objects, '00' bytes without any meaning may occur (for example, due to erased or modified TLV-coded data objects).

B2 Coding of the Length Field of BER-TLV Data Objects

When bit b8 of the most significant byte of the length field is set to 0, the length field consists of only one byte. Bits b7 to b1 code the number of bytes of the value field. The length field is within the range 1 to 127.

When bit b8 of the most significant byte of the length field is set to 1, the subsequent bits b7 to b1 of the most significant byte code the number of subsequent bytes in the length field. The subsequent bytes code an integer representing the number of bytes in the value field. Two bytes are necessary to express up to 255 bytes in the value field.

B3 Coding of the Value Field of Data Objects

A data element is the value field (V) of a primitive BER-TLV data object. A data element is the smallest data field that receives an identifier (a tag). A primitive data object is structured as illustrated in Figure 16:



Figure 16: Primitive BER-TLV Data Object (Data Element)

A constructed BER-TLV data object consists of a tag, a length, and a value field composed of one or more BER-TLV data objects. A record in an AEF governed by this specification is a constructed BER-TLV data object. A constructed data object is structured as illustrated in Figure 17:



Figure 17: Constructed BER-TLV Data Object

--------------------------------------------------------------------------------------------------------------------------------------

Basing on above Constructed BER-TLV specification, I design the TLV to be a linked data structure.

Define Tlv\_t as

typedef struct{

uint16\_t nTag; //Tag field with length up to 2 bytes

uint32\_t nLength; //Length field with length up to 3 bytes, indicating up to 65535 bytes in following Value field

void\* pValue; //Value field pointer

void\* pChild; //pointer pointing to the child TLV of next level

void\* pNext; //pointer pointing to next TLV on the same level

} Tlv\_t;

This is to use a linked data structure to represent TLV object as a TLV node pointing to TLV at child level and also pointing to next TLV on the same level. So when I do “search TLV” and “free TLV”, the code will go through all the TLV nodes. An example is illustrated below:

