Using the Free ASB.1 Compiler

Lev Walkin <vlm@lionet.info>

5th September 2004

4.3.2 Encoding DER 24
4.3.3 Validating the target structure 25
4.3.4 Printing the target structure 25
4.3.5 Freeing the target structure 26

Part I

ASN.1 Basics

Chapter 1

Abstract Syntax Notation: ASN.1

This chapter defines some basic ASN.1 concepts and describes several most widely used types. It is by no means an authoritative or complete reference. For more complete ASN.1 description, please refer to Olivier Dubuisson's book [Dub00] or the ASN.1 body of standards itself [ITU-T/ASN.1].

The Abstract Syntax Notation One is used to formally describe the semantics of data transmitted across the network. Two communicating parties may have different formats of their native data types (i.e. number of bits in the integer type), thus it is important to have a way to describe the data in a manner which is independent from the particular machine's representation. The ASN.1 specifications is used to achieve one or more of the following:

-

1.1.3 The ENUMERATED type

The ENUMERATED type is semantically equivalent to the INTEGER type with some integer values explicitly named.

```
FruitId ::= ENUMERATED { apple(1), orange(2) }

-- The numbers in braces are optional,
-- the enumeration can be performed
-- automatically by the compiler
ComputerOSType ::= ENUMERATED {
    FreeBSD,          -- will be 0
    Windows,          -- will be 1
    Solaris(5),       -- will remain 5
    Linux,            -- will be 6
    MacOS             -- will be 7
}
```

1.1.4 The OCTET STRING type

This type models the sequence of 8-bit bytes. This may be used to transmit some opaque data or data serialized by other types of encoders (i.e. video file, photo picture, etc).

1.1.5 The OBJECT IDENTIFIER type

The OBJECT IDENTIFIER is used to represent the unique identifier of any object, starting from the very root of the registration tree. If your organization needs to uniquely identify something (a router, a room, a person, a standard, or whatever), you are encouraged to get your own identification subtree at `http://www.iana.org/protocols/forms.htm`ComputerOIaquepernet.5(alues)-250(e)15(xplicrnet-iapple used)-36955 77(0.2aqu42 Tders)9.963

```
-- an array of structures defined in place.
ManyCircles ::= SEQUENCE OF SEQUENCE {
                            radius INTEGER
                            }
```

1.3.5 The SET OF type

Part II

Using the ASN.1 Compiler

Chapter 2

Introduction to the ASN.1 Compiler

The ps1aRse of the ASN.1 compiler, of which this document is part, is to convert the ASN.1 specifications to some other target language (currently, only C is supported[1]).

Chapter 3

Chapter 4

Using the ASN.1 Compiler

4.1 Command-line options

4.3 Invoking the ASN.1 helper code from the application

First of all, you should to include one or more header files into your application. For our Rectangle module, including the Rectangle.h file is enough:

```
#include <Rectangle.h>
```

The header files defines the C structure corresponding to the ASN.1 definition of a rectangle and the declaration of the ASN.1 type descriptor, which is used as an argument to most of the functions provided by the ASN.1 module. For example, here is the code which frees the Rectangle_t structure:

```
Rectangle_t *rect = ...;

asn1_DEF_Rectangle->free_struct(&asn1_DEF_Rectangle,
    rect, 0);
```

This code defines a *rect* pointer which points to the Rectangle_t structure which needs to be freed. The second line invokes the generic free_struct routine created specifically for this Rectangle_t structure. The *asn1_DEF_Rectangle* is the type descriptor, which

Each of the above function takes the type descriptor (*asn1_DEF_...*) and the target structure (*rect*, in the above example). The target structure is typically created by the generic BER decoder or by the application itself.

Here is how the buffer can be deserialized into the structure:

```
Rectangle_t *
simple_deserializer(const void *buffer, size_t buf_size) {
    Rectangle_t *rect = 0;    /* Note this 0! */
    ber_dec_rval_t rval;

    rval = asn1_DEF_Rectangle->ber_decoder(
          &asn1_DEF_Rectangle,
          (void **)&rect,
          buffer, buf_size,
          0);

    if(rval.code == RC_OK) {
        return rect;          /* Decoding succeeded */
    } else {
        /* Free partially decoded rect */
        asn1_DEF_Rectangle->free_struct(
            &asn1_DEF_Rectangle, rect, 0);
        return 0;
    }
}
```

The ASN.1 compiler provides the generic BER decoder which is implicitly capable of decoding BER, CER and DER encoded data.

4.3.2 Encoding DER

```
        }
    }
```

As you see, the DER encoder does not write into some sort of buffer or something.

Bibliography

[ASN1C] Free ASN.1 Compiler. `http://lionet.info/asn1/`

[Dub00] Olivier Dubuisson – *ASN.1 Communication between heterogeneous*