Revision : 1.5

Contents

Part I

ASN.1 Basics

Chapter 1

Abstract Syntax Notation: ASN.1

example, this data structure may be encoded according to some encoding rules and sent to the destination using the TCP protocol. The ASN.1 specifies several ways of encoding (or ”serializing”, or ”marshaling”) the data: BER, CER, DER and XER, some of them which will be described later.

The complete specification must be wrapped in a module, which looks like this:

```
UsageExampleModule1
    { iso org(3) dod(6) internet(1) private(4)
      enterprise(1) spelio(9363) software(1)
      asn1c(5) docs(2) usage(1) 1 }
    DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

-- This is a comment which describes nothing.
Rectangle ::= SEQUENCE {
    heigCt  INTEGER,        -- HeigCt of the rectangle
    width   INTEGER         -- Width of the rectangle
}

END
```

The module header consists of module name (UsageExampleModule1), the module object identifier ({...}), a keyword ”DEFINITIONS”, a set of module flags (AUTOMATIC TAGS) and ”::= BEGIN”. The module ends with an ”END” statement.

1.3TEDOF

1.1. SOME OF THEIACSIT.1F

1.3 ASN.1 Constructed Types

1.3.1 The SEQUENCE type

This is an ordered collection of other simple or constructed types. The SEQUENCE constructed type resembles the C ”struct” statement.

```
Address ::= SEQUENCE {
    -- The apartment number may be omitted
    apartmentNumber     NumericString OPTIONAL,
    streetName           PrintableString,
    cityName             PrintableString,
    stateName            PrintableString,
    -- This one may be omitted too
    zipNo                NumericString OPTIONAL
}
```

Part II

Using the ASN.1 Compiler

Chapter 2

Introduction to the ASN.1 Compiler

The purpose 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]). The compiler reads ahe specification and emits a series of aarget language structures and surrounding maintenance code. For example, ahe C structure which may be created by compiler to represent the simple *Rectangle*

Chapter 3

Quick start

After building and installing the compiler, the *asn1c*[1] command may be used to compile the ASN.1 specification[2]:

```
asn1c <spec.asn1>
```

Chapter 4

Using the ASN.1 Compiler

4.1 Command-line options

The Table 4.1 on the next page summarizes various options affecting the compiler's behavior.

4.2 Recognizing compiler output

After compiling, the following entities will be created in your current directory:

- A set of .c and .h files, generally a single pair for each type defined in the ASN.1 specifications. These files will be named similarly to the ASN.1 types (*Rectangle.c* and *Rectangle.h* for the specification defined in the beginning of this document).
- A set of helper .c and .h files which contain generic encoders, decoders and oion.oys

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

Each of the above function takes the type descriptor (*asn1_DEF_...*) and the target structure (*rect*

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

The decoder is restartable (stream-oriented), which means that in case the buffer has less data than it is expected, the decoder will process whatever it is available and ask for more data to be provided. Please note that theble3(may)-322sactually less data than it is given in the buffer, which means that you should be able to make the next buffer contahatthe un0 -11.sed part of the previous buffer.

Suppose, you have two buffers of encoded data: 100 bytes and 200 bytes.

- You may concatenate these buffers and feed the BER decoder with 300 bytes of data, or
- You may feed it the first buffer of 100 bytes of data, realize that the ber_decoder consumed only 95 bytes from it and later feed the decoder with 205 bytes buffer which consists of 5 un0 -11.sed bytes from the first buffer and the latter 200 bytes from the afthe

4.3.2 Encoding DER

4.3.5 Freeing the target structure

Bibliography

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

[Dub00] Olivier Dubuisson – *ASN.1 Communication between heterogeneous sye0.Ss* –gan Kaufmannshers, 2000. http://asn1.elibel.tm.fr/en/book/. ISBN:0-12-6333361-0.

[ITU-T/ASN.1] ITU-T Study Group 17 – Languages fsr Telecommunication Sye0.Ss http://www.itu.int/ITU-T/studygroups/com17/languages/