Using the Open Source ASN.1 Compiler

Chapter 1

Introduction to the ASN.1 Compiler

The purpose of the ASN.1 compiler, of which t
specifications in ASN.1 notation into some oth
and C++ target languages are supported, the la

The compiler reads the specification and emits

It would also create the code for converting this structure into platform-independent wire representation (a serializer API) and the decoder of such wire representation back into local, machine-specific type (a deserializer API).

1.1 Quick start with asn1c

After building and installing the compiler, the *asn1c*[3]

Language Options	**Description**
-fall-defs-global	Normally the compiler hides the definitions (asn_DEF_xxx)

der_encoder This is the generio0(i1(DA)1250(Ti1(_encoder)-270(DistinguiA)1shed0(Ti1(EencoinA)1g0(Ti1(R

- 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 unprocessed bytes from the first buffer and the additional

```
rval = ber_decode(0, &asn_DEF_Rectangle, (void **)&rect,
    buffer, buf_size);
```

```
        er = der_encode(&asn_DEF_Rect, rect,
            write_stream, ostream);
        if(er.encoded == -1) {
            /*
             * Failed to encode the rectangle data.
             */
            fprintf(stderr, "Cannot encode %s: %s\n",
                er.failed_type->name,
                strerror(errno));
            return -1;
        } else {
            /* Return the number of bytes */
            return er.encoded;
        }
    }
```

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

```
 */
int
print_as_XML(FILE *ostream, Rectangle_t *rect) {
    asn_enc_rval_t er;  /* Encoder return value */
```

```
#include <stdio.h>
#include <sys/types.h>
#include <Rectangle.h>   /* Rectangle ASN.1 type  */

/*
 * This is a custom function which writes the
 * encoded output into some FILE stream.
 */
static int
```

```
  if(ec.encoded == -1) {
    fprintf(stderr,
      "Could not encode Rectangle (at %s)\n",
      ec.failed_type ? ec.failed_type->name : "unknown");
    exit(65); /* better, EX_DATAERR */
  } else {
    fprintf(stderr, "Created %s with BER encoded Rectangle\n",
      filename);
  }
}

/* Also print the constructed Rectangle XER encoded (XML) */
xer_fprint(stdout, &asn_DEF_Rectangle, rectangle);
```

3.2 A ”Rectangle” Decoder

This example will help you to create a simple BER decoder of a simple ”Rectangle” type used throughout this document.

1.

6. Compile all files together using C compiler (varies by platform):

```
cc -I. -o rdecode *.c
```

7. Voila! You have just created the BER decoder of a Rectangle type, named **rdecode**!

Chapter 4

Constraint validation examples

This chapter shows how to define ASN.1 constraints and use the generated validation code.

4.1 Adding constraints into ”Rectangle” type

```
int ret;           /* Return value */
char errbuf[128];  /* Buffer for error message */
size_t errlen = sizeof(errbuf);  /* Size of the buffer */

/* ... here may go Rectangle decoding code ... */

ret = asn_check_constraints(&asn_DEF_Rectangle,
```

s;

Chapter 5

Abstract Syntax Notation: ASN.1

This chapter defines some basic ASN.1 concepts and describes several most widely used types. It is by no meacs an authoritative or complete reference. F(or)-333(mor)37(e)-334(complete)]TJ 0 -11.956 Td[(ASN.

5.1.3 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,          -- acquires value 0
    Windows,          -- acquires value 1
    Solaris(5),       -- remains 5
    Linux,            -- becomes 6
    MacOS             -- becomes 7
}
```

5.1.4 The OCTET STRING type

This type models the sequence of 8-bit bytes. This may be used to transmit some

5.3 ASN.1 Constructed Types

5.3.1 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
}
```

5.3.2 The SET type

This is a collection of other simple or constructed types. Ordering is not important. The data may arrive in the order which is different from the order of specification. Data is encoded in the order not necessarily corresponding to the order of specification.

5.3.3 The CHOICE type

This type is just a choice between the subtypes specified in it. The CHOICE type

Bibliography

[ASN1C] The Open Source ASN.1 Compiler. http://lionet.info/asn1c

[AONL] Online ASN.1 Compiler. http://lionet.info/asn1c/asn1c.cgi

[Dub00] Olivier Dubuisson — *ASN.1 Communication between heterogeneous systems* — Morgan Kaufmann Publishers, 2000. http://asn1.elibel.tm.fr/en/book/. ISBN:0-12-6333361-0.

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