Contents

ASN.1 Basics

Chapter 1

Abstract Syntax Notation: ASN.1

ASN.1. For 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

1.1.3 The ENUMERATED type

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
}
```

1.3.2 The SET type

This is a collection of other simple or constructed types. Ordering is not important. The

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

1.3.5 The SET OF type

The SET OF type models the bag of structures. It resembles the SEQUENCE OF type, but the order is not important: i.e. the elements may arrive in the order which is not

Part II

ASN.1 Compiler

Chapter 2

Introduction to the ASN.1 Compiler

Chapter 3

Quick start

After building and installing the compi9er, the*asn1c*[1]

Chapter 4

Overall Options	Description
-E	Stop after the parsing stage and print the reconstructed ASN.1 specification code to the standard output.
-F	Used together with -E, instructs the compiler to stop after the ASN.1 syntax tree fixing stage and dump the reconstructed ASN.1 specification to the standard output.
-P	Dump the compiled output to the standard output instead of

4.3.2 Encoding DER

The Distinguished Encoding Rules is the *canonical* variant of BER encoding rules. The

Please look into der_encoder.h for the precise definition of der_encode() and related types.

4.3.3 Encoding XER

The XER stands for XML Encoding Rules, where XML, in turn, is eXtensible Markup

it does not point to the memory block directly allocated by memory allocation routine, but instead lies within such a block allocated for my_figure structure.

To solve this problem, the free_struct 64pTIa tem,gume t(besidd

Part III

Examples

```
#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
write_out(const void *buffer, size_t size, void *app_key) {
    FILE *out_fp = app_key;
    size_t wrote;

    wrote = fwrite(buffer, 1, size, out_fp);

    return (wrote == size) ? 0 : -1;
}

int main(int ac, char **av) {
    Rectangle_t *rectangle; /* Type to encode        */
    asn_enc_rval_t ec;      /* Encoder return value  */
```

5.2 A ”Rectangle” Decoder

This example will help you to create a simple BER decoder of a simple ”Rectangle”

Chapter 6

Constraint validation examples

This chapter shows how to de,002ne ASN.1 constraints and use the generated validation code.

6.1 Adding constraints into ”Rectangle” type

This example shows how to add basic constraints to the ASN.1 speci,002cation and how to invoke the constraints validation code in your application.

1. Create a file named **rectangle.asn1** with the following contents:

```
RectangleModuleWith/F30 raints DEFINITIONS ::=
BEGIN

Rectangle ::= SEQUENCE {
    height  INTEGER (0..100), -- Value range constraint
    width   INTEGER (0..MAX)  -- Makes width non-negative
}

END
```

2. Compile the file according o procedures shown in the previous chapter.

3. Modify the Rectangle type processing routine (you can start with the main() routine shown in the Section 5.2 on page 34) by placing the following snippet of code *before* encoding and/or *after* decoding the Rec[1]tangle type

[1] Placing the constraint checking code *before* encoding helps to make sure you know the data is correct and within constraints before sharing the data with anyone else.

Placing the constraint checking code *after* decoding, but before any further action depending on the decoded data, helps to make sure the application got the valid contents before making use of it.

```
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,
        rectangle, errbuf, &errlen);
/* assert(errlen ;5e600(hexgle,)-r600(91(aintsi(aintsyou...)-600(here)-rel(
```

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/