ASN.1 structures parser

This is part of the GnuTLS project

Contents

Chapter 1

ASN.1 structures handling

1.1 Introduction

This document describes the version 0.2.4 of library 'libtasn1' developed for ASN1 (Abstract Syntax Notation One) structures management. The main features of this library are:

- on line ASN1 structure management that doesn't require any C code file generation.
- off line ASN1 structure management with C code file generation containing an array.
- DER (Distinguish Encoding Rules) encoding
- no limits for INTEGER and ENUMERATED values

1.2 ASN.1 syntax

The parser is case sensitive. The comments begin with "– " and end at the end of lines. An example is in "pkix.asn" file. ASN.1 definitions must have this syntax:

```
definitions_name {<object definition>}

DEFINITIONS <EXPLICIT or IMPLICIT> TAGS ::=

BEGIN

<type and constants definitions>

END
```

The token "::=" must be separate from others elements, so this is a wrong declaration: Version ::=INTEGER the correct one is : Version ::= INTEGER Here is the list of types that the parser can manage:

- INTEGER
- ENUMERATED
- BOOLEAN
- OBJECT IDENTIFIER
- NULL
- BIT STRING
- OCTET STRING
- UTCTime
- GeneralizedTime
- GeneralString
- SEQUENCE
- SEQUENCE OF
- SET
- SET OF
- CHOICE
- ANY
- ANY DEFINED BY

This version doesn't manage REAL type. It doesn't allow the "EXPORT" and "IMPORT" sections too.

The SIZE constraints are allowed, but no check is done on them.

1.3 Naming

With this definitions:

```
Example { 1 2 3 4 }

DEFINITIONS EXPLICIT TAGS ::=
```

```
BEGIN

Group ::= SEQUENCE {
   id   OBJECT IDENTIFIER,
   value  Value
}

Value ::= SEQUENCE {
   value1  INTEGER,
   value2  BOOLEAN
}

END
```

to identify the type 'Group' you have to use the null terminated string "Example.Group". Others examples: Field 'id' in 'Group' type : "Example.Group.id" Field 'value1' in field 'value' in type 'Group': "Example.Group.value.value1" These strings are used in functions that are described below. Elements of structured types that don't have a name, receive the name "?1","?2", and so on. The name "?LAST" indicates the last element of a SET_OF or SEQUENCE_OF.

1.4 Library Notes

The header file of this library is libtasn1.h . The main type used in it is ASN1_TYPE, and it's used to store the ASN1 definitions and structures (instances). The constant ASN1_TYPE_EMPTY can be used for the variable initialization.

Example: ASN1_TYPE definitions=ASN1_TYPE_EMPTY;

Some functions require a parameter named errorDescription of char* type. The array must be already allocated and must have at least MAX_ERROR_DESCRIPTION_SIZE bytes (E.g: char Description[MAX_ERROR_DESCRIPTION_SIZE];).

MAX_NAME_SIZE indicates the maximum number of characters of a name inside a file with ASN1 definitions.

1.5 Future developments

1. add functions for a C code file generation containing equivalent data structures (not a single array like now).
2. type REAL

Chapter 2

Utilities

2.1 asn1Parser

asn1Parser reads one file with ASN1 definitions and generates a file with an array to use with libasn1 functions.

Usage: asn1Parser [options] file

Options:

- -h : shows the help message.
- -v : shows version information and exit.
- -c : checks the syntax only.
- -o file : output file.
- -n name : array name.

2.2 asn1Coding

asn1Coding generates a DER encoding from a file with ASN1 definitions and another one with assignments. The file with assignments must have this syntax:

InstanceName Asn1Definition

nameString value

nameString value

...

The output file is a binary file with the DER encoding.

Usage: asn1Coding [options] file1 file2

- file1 : file with ASN1 definitions.

- file2 : file with assignments.

Options:

- -h : shows the help message.
- -v : shows version information and exit.
- -c : checks the syntax only.
- -o file : output file.

2.3 asn1Decoding

asn1Decoding generates an ASN1 structure from a file with ASN1 definitions and a binary file with a DER encoding.

Usage: asn1Decoding [options] file1 file2 type

- file1 : file with ASN1 definitions.
- file2 : binary file with a DER encoding.
- type : ASN1 definition name.

Options:

- -h : shows the help message.
- -v : shows version information and exit.
- -c : checks the syntax only.
- -o file : output file.

Chapter 3

Function reference

3.0.1 asn1_parser2tree

asn1_retCode **asn1_parser2tree** (*const char ** **file_name** , *ASN1_TYPE ** **definitions** , *char ** **errorDescription**)

Arguments

- *const char ** **file_name**: specify the path and the name of file that contains ASN.1 declarations.
- *ASN1_TYPE ** **definitions**: return the pointer to the structure created from "file_name" ASN.1 declarations.
- *char ** **errorDescription**: return the error description or an empty string if success.

Description

Creates the structures needed to manage the definitions included in *FILE_NAME file.

Returns

ASN1_SUCCESS: the file has a correct syntax and every identifier is known.

ASN1_ELEMENT_NOT_EMPTY: *POINTER not ASN1_TYPE_EMPTY.

ASN1_FILE_NOT_FOUND: an error occured while opening FILE_NAME.

ASN1_SYNTAX_ERROR: the syntax is not correct.

ASN1_IDENTIFIER_NOT_FOUND: in the file there is an identifier that is not defined. ASN1_NAME_TOO_LONG: in the file there is an identifier whith more than MAX_NAME_SIZE characters.

3.0.2 asn1_parser2array

int **asn1_parser2array** (*const char ** **inputFileName** , *const char ** **output-**

FileName , *const char ** **vectorName** , *char ** **errorDescription**)

Arguments

- *const char ** **inputFileName**: specify the path and the name of file that contains ASN.1 declarations.
- *const char ** **outputFileName**: specify the path and the name of file that will contain the C vector definition.
- *const char ** **vectorName**: specify the name of the C vector.
- *char ** **errorDescription**: return the error description or an empty string if success.

Description

Creates a file containing a C vector to use to manage the definitions included in *INPUTFILENAME file. If *INPUTFILENAME is "/aa/bb/xx.yy" and OUTPUTFILENAME is NULL, the file created is "/aa/bb/xx_asn1_tab.c". If VECTORNAME is NULL the vector name will be "xx_asn1_tab".

Returns

ASN1_SUCCESS: the file has a correct syntax and every identifier is known.

ASN1_FILE_NOT_FOUND: an error occured while opening FILE_NAME.

ASN1_SYNTAX_ERROR: the syntax is not correct.

ASN1_IDENTIFIER_NOT_FOUND: in the file there is an identifier that is not defined. ASN1_NAME_TOO_LONG: in the file there is an identifier whith more than MAX_NAME_SIZE characters.

3.0.3 asn1_der_decoding

asn1_retCode **asn1_der_decoding** (*ASN1_TYPE ** **element** , *const unsigned char ** **der** , *int* **len** , *char ** **errorDescription**)

Arguments

- *ASN1_TYPE ** **element**: pointer to an ASN1 structure
- *const unsigned char ** **der**: vector that contains the DER encoding.
- *int* **len**: number of bytes of *der: der[0]..der[len-1]
- *char ** **errorDescription**:

Description

Fill the structure *ELEMENT with values of a DER encoding string. The sructure must just be created with function 'create_stucture'. If an error accurs during de decoding procedure, the *ELEMENT is deleted and set equal to ASN1_TYPE_EMPTY.

Returns

ASN1_SUCCESS: DER encoding OK

ASN1_ELEMENT_NOT_FOUND: ELEMENT is ASN1_TYPE_EMPTY.

ASN1_TAG_ERROR,ASN1_DER_ERROR: the der encoding doesn't match the structure NAME. *ELEMENT deleted.

3.0.4 asn1_der_decoding_element

asn1_retCode **asn1_der_decoding_element** (*ASN1_TYPE ** **structure** , *const char ** **elementName** , *const unsigned char ** **der** , *int* **len** , *char ** **errorDescription**)

Arguments

- *ASN1_TYPE ** **structure**: pointer to an ASN1 structure
- *const char ** **elementName**: name of the element to fill
- *const unsigned char ** **der**: vector that contains the DER encoding of the whole structure.
- *int* **len**: number of bytes of *der: der[0]..der[len-1]
- *char ** **errorDescription**: null-terminated string contains details when an arror accured.

Description

Fill the element named ELEMENTNAME with values of a DER encoding string. The sructure must just be created with function 'create_stucture'. The DER vector must contain the encoding string of the whole STRUCTURE. If an error accurs during the decoding procedure, the *STRUCTURE is deleted and set equal to ASN1_TYPE_EMPTY.

Returns

ASN1_SUCCESS: DER encoding OK

ASN1_ELEMENT_NOT_FOUND: ELEMENT is ASN1_TYPE_EMPTY or elementName == NULL.

ASN1_TAG_ERROR,ASN1_DER_ERROR: the der encoding doesn't match the structure STRUCTURE. *ELEMENT deleted.

3.0.5 asn1_der_decoding_startEnd

asn1_retCode **asn1_der_decoding_startEnd** (*ASN1_TYPE* **element** , *const unsigned char ** **der** , *int* **len** , *const char ** **name_element** , *int ** **start** , *int ** **end**)

Arguments

- *ASN1_TYPE* **element**: pointer to an ASN1 element
- *const unsigned char ** **der**: vector that contains the DER encoding.
- *int* **len**: number of bytes of *der: der[0]..der[len-1]
- *const char ** **name_element**: an element of NAME structure.
- *int ** **start**: the position of the first byte of NAME_ELEMENT decoding (der[*start])
- *int ** **end**: the position of the last byte of NAME_ELEMENT decoding (der[*end])

Description

Find the start and end point of an element in a DER encoding string. I mean that if you have a der encoding and you have already used the function "asn1_der_decoding" to fill a structure, it may happen that you want to find the piece of string concerning an element of the structure.

Example

the sequence "tbsCertificate" inside an X509 certificate.

Returns

ASN1_SUCCESS: DER encoding OK

ASN1_ELEMENT_NOT_FOUND: ELEMENT is ASN1_TYPE EMPTY or NAME_ELEMENT is not a valid element.

ASN1_TAG_ERROR,ASN1_DER_ERROR: the der encoding doesn't match the structure ELEMENT.

3.0.6 asn1_expand_any_defined_by

asn1_retCode **asn1_expand_any_defined_by** (*ASN1_TYPE* **definitions** , *ASN1_TYPE ** **element**)

Arguments

- *ASN1_TYPE* **definitions**: ASN1 definitions
- *ASN1_TYPE ** **element**: pointer to an ASN1 structure

Description

Expands every "ANY DEFINED BY" element of a structure created from a DER decoding process (asn1_der_decoding function). The element ANY must be defined by an OBJECT IDENTIFIER. The type used to expand the element ANY is the first one following the definition of the actual value of the OBJECT IDENTIFIER.

Description

Expands every "ANY DEFINED BY" element of a structure created from a DER decoding process (asn1_der_decoding function). The element ANY must be defined by an OBJECT IDENTIFIER. The type used to expand the element ANY is the first one following the definition of the actual value of the OBJECT IDENTIFIER.

Returns

ASN1_SUCCESS: substitution OK

ASN1_ERROR_TYPE_ANY: some "ANY DEFINED BY" element couldn't be expanded due to a problem in OBJECT_ID → TYPE association. other errors: result of der decoding process.

3.0.7 asn1_expand_octet_string

asn1_retCode **asn1_expand_octet_string** (*ASN1_TYPE* **definitions** , *ASN1_TYPE* * **element** , *const char* * **octetName** , *const char* * **objectName**)

Arguments

- *ASN1_TYPE* **definitions**: ASN1 definitions
- *ASN1_TYPE* * **element**: pointer to an ASN1 structure
- *const char* * **octetName**: name of the OCTECT STRING field to expand.
- *const char* * **objectName**: name of the OBJECT IDENTIFIER field to use to define the type for expansion.

Description

Expands an "OCTET STRING" element of a structure created from a DER decoding process (asn1_der_decoding function). The type used for expansion is the first one following the definition of the actual value of the OBJECT IDENTIFIER indicated by OBJECTNAME.

Description

Expands an "OCTET STRING" element of a structure created from a DER decoding process (asn1_der_decoding function). The type used for expansion is the first one following the definition of the actual value of the OBJECT IDENTIFIER indicated by OBJECTNAME.

Returns

ASN1_SUCCESS: substitution OK

ASN1_ELEMENT_NOT_FOUND: OBJECTNAME or OCTETNAME are not correct.

ASN1_VALUE_NOT_VALID: wasn't possible to find the type to use for expansion.

other errors: result of der decoding process.

3.0.8 libtasn1_perror

void **libtasn1_perror** (*asn1_retCode* **error**)

Arguments

- *asn1_retCode* **error**: is an error returned by a libasn1 function.

Description

This function is like perror(). The only difference is that it accepts an error returned by a libasn1 function.

3.0.9 libtasn1_strerror

*const char** **libtasn1_strerror** (*asn1_retCode* **error**)

Arguments

- *asn1_retCode* **error**: is an error returned by a libtasn1 function.

Description

This function is similar to strerror(). The only difference is that it accepts an error (number) returned by a libasn1 function.

3.0.10 asn1_check_version

*const char ** **asn1_check_version** (*const char ** **req_version**)

Arguments

- *const char ** **req_version**: the version to check

Description

Check that the the version of the library is at minimum the requested one and return the version string; return NULL if the condition is not satisfied. If a NULL is passed to this function, no check is done, but the version string is simply returned.

3.0.11 asn1_array2tree

asn1_retCode **asn1_array2tree** (*const ASN1_ARRAY_TYPE ** **array** , *ASN1_TYPE ** **definitions** , *char ** **errorDescription**)

Arguments

- *const ASN1_ARRAY_TYPE ** **array**: specify the array that contains ASN.1 declarations

- *ASN1_TYPE ** **definitions**: return the pointer to the structure created by *ARRAY ASN.1 declarations

- *char ** **errorDescription**: return the error description.

Description

Creates the structures needed to manage the ASN1 definitions. ARRAY is a vector created by 'asn1_parser_asn1_file_c' function.

Returns

ASN1_SUCCESS: structure created correctly.

ASN1_ELEMENT_NOT_EMPTY: *DEFINITIONS not ASN1_TYPE_EMPTY

ASN1_IDENTIFIER_NOT_FOUND: in the file there is an identifier that is not defined (see ERRORDESCRIPTION for more information).

ASN1_ARRAY_ERROR: the array pointed by ARRAY is wrong.

3.0.12 asn1_delete_structure

asn1_retCode **asn1_delete_structure** (*ASN1_TYPE ** **structure**)

Arguments

- *ASN1_TYPE ** **structure**: pointer to the structure that you want to delete.

Description

Deletes the structure *ROOT. At the end *ROOT is setted to ASN1_TYPE_EMPTY.

Returns

ASN1_SUCCESS: everything OK

ASN1_ELEMENT_NOT_FOUND: *root==ASN1_TYPE_EMPTY.

3.0.13 asn1_create_element

asn1_retCode **asn1_create_element** (*ASN1_TYPE* **definitions** , *const char ** **source_name** , *ASN1_TYPE ** **element**)

Arguments

- *ASN1_TYPE* **definitions**: pointer to the structure returned by "parser_asn1" function

- *const char ** **source_name**: the name of the type of the new structure (must be inside p_structure).

- *ASN1_TYPE ** **element**: pointer to the structure created.

Description

Creates a structure called DEST_NAME of type SOURCE_NAME.

Returns

ASN1_SUCCESS: creation OK

ASN1_ELEMENT_NOT_FOUND: SOURCE_NAME isn't known

Example

using "pkix.asn" result=asn1_create_structure(cert_def,"PKIX1.Certificate",*cert*);

3.0.14 asn1_print_structure

void **asn1_print_structure** (*FILE ** **out** , *ASN1_TYPE* **structure** , *const char ** **name** , *int* **mode**)

Arguments

- *FILE ** **out**: pointer to the output file (e.g. stdout).
- *ASN1_TYPE* **structure**: pointer to the structure that you want to visit.
- *const char ** **name**: an element of the structure
- *int* **mode**:

Description

Prints on the standard output the structure's tree starting from the NAME element inside the structure *POINTER.

3.0.15 asn1_number_of_elements

asn1_retCode **asn1_number_of_elements** (*ASN1_TYPE* **element** , *const char ** **name** , *int ** **num**)

Arguments

- *ASN1_TYPE* **element**: pointer to the root of an ASN1 structure.
- *const char ** **name**: the name of a sub-structure of ROOT.
- *int ** **num**: pointer to an integer where the result will be stored

Description

Counts the number of elements of a sub-structure called NAME with names equal to "?1","?2", ...

Returns

ASN1_SUCCESS: creation OK ASN1_ELEMENT_NOT_FOUND: NAME isn't known ASN1_GENERIC_ERROR: pointer num equal to NULL

3.0.16 asn1_find_structure_from_oid

*const char** **asn1_find_structure_from_oid** (*ASN1_TYPE* **definitions** , *const char ** **oidValue**)

Arguments

- *ASN1_TYPE* **definitions**: ASN1 definitions
- *const char ** **oidValue**: value of the OID to search (e.g. ”1.2.3.4”).

Description

Search the structure that is defined just after an OID definition.

Description

Search the structure that is defined just after an OID definition.

Returns

NULL when OIDVALUE not found,

otherwise the pointer to a constant string that contains the element name defined just after the OID.

3.0.17 asn1_write_value

asn1_retCode **asn1_write_value** (*node_asn ** **node_root** , *const char ** **name** , *const unsigned char ** **value** , *int* **len**)

Arguments

- *node_asn ** **node_root**: pointer to a structure
- *const char ** **name**: the name of the element inside the structure that you want to set.
- *const unsigned char ** **value**: vector used to specify the value to set. If len is ¿0, VALUE must be a two's complement form integer. if len=0 *VALUE must be a null terminated string with an integer value.
- *int* **len**: number of bytes of *value to use to set the value: value[0]..value[len-1] or 0 if value is a null terminated string

Description

Set the value of one element inside a structure.

Returns

ASN1_SUCCESS: set value OK

ASN1_ELEMENT_NOT_FOUND: NAME is not a valid element.

ASN1_VALUE_NOT_VALID: VALUE has a wrong format.

Examples

description for each type

- INTEGER: VALUE must contain a two's complement form integer. value[0]=0xFF , len=1 → integer=-1 value[0]=0xFF value[1]=0xFF , len=2 → integer=-1 value[0]=0x01 , len=1 → integer= 1 value[0]=0x00 value[1]=0x01 , len=2 → integer= 1 value="123" , len=0 → integer= 123
- ENUMERATED: as INTEGER (but only with not negative numbers)
- BOOLEAN: VALUE must be the null terminated string "TRUE" or "FALSE" and LEN != 0 value="TRUE" , len=1 → boolean=TRUE value="FALSE" , len=1 → boolean=FALSE
- OBJECT IDENTIFIER: VALUE must be a null terminated string with each number separated by a dot (e.g. "1.2.3.543.1"). LEN != 0 value="1 2 840 10040 4 3" , len=1 → OID=dsa-with-sha
- UTCTime: VALUE must be a null terminated string in one of these formats: "YYMMDDhhmmssZ" "YYMMDDhhmmssZ" "YYMMDDhhmmss+hh'mm'" "YYMMDDhhmmss-hh'mm'" "YYMMDDhhmm+hh'mm'" "YYMMDDhhmm-hh'mm'". LEN != 0 value="9801011200Z" , len=1 → time=Jannuary 1st, 1998 at 12h 00m Greenwich Mean Time
- GeneralizedTime: VALUE must be in one of this format: "YYYYMMDDhhmmss.sZ" "YYYYMMDDhhmmss.sZ" "YYYYMMDDhhmmss.s+hh'mm'" "YYYYMMDDhhmmss.s-hh'mm'" "YYYYMMDDhhmm+hh'mm'" "YYYYMMDDhhmm-hh'mm'" where ss.s indicates the seconds with any precision like "10.1" or "01.02". LEN != 0 value="2001010112001.12-0700" , len=1 → time=Jannuary 1st, 2001 at 12h 00m 01.12s Pacific Daylight Time
- OCTET STRING: VALUE contains the octet string and LEN is the number of octet. value="\x01\x02\x03" , len=3 → three bytes octet string
- GeneralString: VALUE contains the generalstring and LEN is the number of octet. value="\x01\x02\x03" , len=3 → three bytes generalstring
- BIT STRING: VALUE contains the bit string organized by bytes and LEN is the number of bits. value="\xCF" , len=6 → bit string="110011" (six bits)
- CHOICE: if NAME indicates a choice type, VALUE must specify one of the alternatives with a null terminated string. LEN != 0 Using "pkix.asn": result=asn1_write_value(cert,"certificate1.tbsCertificate.subject","rdnSequence",1);
- ANY: VALUE indicates the der encoding of a structure. LEN != 0
- SEQUENCE OF: VALUE must be the null terminated string "NEW" and LEN != 0. With this instruction another element is appended in the sequence. The name of this element will be "?1" if it's the first one, "?2" for the second and so on.

  Using "pkix.asn":

  result=asn1_write_value(cert,"certificate1.tbsCertificate.subject.rdnSequence","NEW",1);

- SET OF: the same as SEQUENCE OF. Using "pkix.asn":
  result=asn1_write_value(cert,"tbsCertificate.subject.rdnSequence.?LAST","NEW",1);

If an element is OPTIONAL and you want to delete it, you must use the value=NULL and len=0.

Using "pkix.asn":

result=asn1_write_value(cert,"tbsCertificate.issuerUniqueID",NULL,0);

3.0.18 asn1_read_value

asn1_retCode **asn1_read_value** (*node_asn ** **root** , *const char ** **name** , *unsigned char ** **value** , *int ** **len**)

Arguments

- *node_asn ** **root**: pointer to a structure
- *const char ** **name**: the name of the element inside a structure that you want to read.
- *unsigned char ** **value**: vector that will contain the element's content. VALUE must be a pointer to memory cells already allocated.
- *int ** **len**: number of bytes of *value: value[0]..value[len-1]. Initialy holds the sizeof value.

Description

Returns the value of one element inside a structure.

Returns

ASN1_SUCCESS: set value OK

ASN1_ELEMENT_NOT_FOUND: NAME is not a valid element.

ASN1_VALUE_NOT_FOUND: there isn't any value for the element selected.

ASN1_MEM_ERROR: the value vector isn't big enough to store the result. In this case LEN will contain the number of bytes needed.

Examples

a description for each type

- INTEGER: VALUE will contain a two's complement form integer. integer=-1 → value[0]=0xFF , len=1 integer=1 → value[0]=0x01 , len=1
- ENUMERATED: as INTEGER (but only with not negative numbers)
- BOOLEAN: VALUE will be the null terminated string "TRUE" or "FALSE" and LEN=5 or LEN=6
- OBJECT IDENTIFIER: VALUE will be a null terminated string with each number separated by a dot (i.e. "1.2.3.543.1"). LEN = strlen(VALUE)+1

- UTCTime: VALUE will be a null terminated string in one of these formats: "YYMMDDhhmmss+hh'mm'" or "YYMMDDhhmmss-hh'mm'" LEN=strlen(VALUE)+1
- GeneralizedTime: VALUE will be a null terminated string in the same format used to set the value
- OCTET STRING: VALUE will contain the octet string and LEN will be the number of octet.
- GeneralString: VALUE will contain the generalstring and LEN will be the number of octet.
- BIT STRING: VALUE will contain the bit string organized by bytes and LEN will be the number of bits.
- CHOICE: if NAME indicates a choice type, VALUE will specify the alternative selected
- ANY: if NAME indicates an any type, VALUE will indicate the DER encoding of the structure actually used.

If an element is OPTIONAL and the function "read_value" returns ASN1_ELEMENT_NOT_FOUND, it means that this element wasn't present in the der encoding that created the structure. The first element of a SEQUENCE_OF or SET_OF is named "?1". The second one "?2" and so on.

3.0.19 asn1_read_tag

asn1_retCode **asn1_read_tag** (*node_asn* * **root** , *const char* * **name** , *int* * **tagValue** , *int* * **classValue**)

Arguments

- *node_asn* * **root**: pointer to a structure
- *const char* * **name**: the name of the element inside a structure.
- *int* * **tagValue**: variable that will contain the TAG value.
- *int* * **classValue**: variable that will specify the TAG type.

Description

Returns the TAG and the CLASS of one element inside a structure.

CLASS can have one of these constants

ASN1_CLASS_APPLICATION, ASN1_CLASS_UNIVERSAL, ASN1_CLASS_PRIVATE or ASN1_CLASS_CONTEXT_SPECIFIC.

Returns

ASN1_SUCCESS: set value OK

ASN1_ELEMENT_NOT_FOUND: NAME is not a valid element.

3.0.20 asn1_der_coding

asn1_retCode **asn1_der_coding** (*ASN1_TYPE* **element** , *const char ** **name** , *unsigned char ** **der** , *int ** **len** , *char ** **ErrorDescription**)

Arguments

- *ASN1_TYPE* **element**: pointer to an ASN1 element
- *const char ** **name**: the name of the structure you want to encode (it must be inside *POINTER).
- *unsigned char ** **der**: vector that will contain the DER encoding. DER must be a pointer to memory cells already allocated.
- *int ** **len**: number of bytes of *der: der[0]..der[len-1], Initialy holds the sizeof of der vector.
- *char ** **ErrorDescription**:

Description

Creates the DER encoding for the NAME structure (inside *POINTER structure).

Returns

ASN1_SUCCESS: DER encoding OK

ASN1_ELEMENT_NOT_FOUND: NAME is not a valid element.

ASN1_VALUE_NOT_FOUND: there is an element without a value.

ASN1_MEM_ERROR: der vector isn't big enough. Also in this case LEN will contain the length needed.

Chapter 4

GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

4.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

4.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

4.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

4.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

4.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

4.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the

Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

4.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

4.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

4.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.